

Documented Code For glossaries v4.29

Nicola L.C. Talbot

Dickimaw Books

<http://www.dickimaw-books.com/>

2017-01-19

This is the documented code for the glossaries package. This bundle comes with the following documentation:

[glossariesbegin.pdf](#) If you are a complete beginner, start with “The glossaries package: a guide for beginners”.

[glossary2glossaries.pdf](#) If you are moving over from the obsolete glossary package, read “Upgrading from the glossary package to the glossaries package”.

[glossaries-user.pdf](#) For the main user guide, read “glossaries.sty v4.29: L^AT_EX2_ε Package to Assist Generating Glossaries”.

[mfirstuc-manual.pdf](#) The commands provided by the mfirstuc package are briefly described in “mfirstuc.sty: uppercasing first letter”.

[glossaries-code.pdf](#) This document is for advanced users wishing to know more about the inner workings of the glossaries package.

INSTALL Installation instructions.

CHANGES Change log.

README Package summary.

The user level commands described in the user manual ([glossaries-user.pdf](#)) may be considered “future-proof”. Even if they become deprecated, they should still work for old documents (although they may not work in a document that also contains new commands introduced since the old commands were deprecated, and you may need to specify a compatibility mode).

The internal commands in *this* document that aren’t documented in the *user manual* should not be considered future-proof and are liable to change. If you want a new user level command, you can post a feature request at <http://www.dickimaw-books.com/feature-request.html>. If you are a package writer wanting to integrate your package with glossaries, it’s better to request a new user level command than to hack these internals.

Contents

1	Main Package Code	4
1.1	Package Definition	4
1.2	Package Options	5
1.3	Predefined Text	30
1.4	Xindy	40
1.5	Loops and conditionals	49
1.6	Defining new glossaries	55
1.7	Defining new entries	60
1.8	Resetting and unsetting entry flags	85
1.9	Keeping Track of How Many Times an Entry Has Been Unset	88
1.10	Loading files containing glossary entries	93
1.11	Using glossary entries in the text	93
1.12	Adding an entry to the glossary without generating text	152
1.13	Creating associated files	154
1.14	Writing information to associated files	172
1.15	Glossary Entry Cross-References	179
1.16	Displaying the glossary	181
1.17	Acronyms	210
1.18	Predefined acronym styles	214
1.19	Predefined Glossary Styles	246
1.20	Debugging Commands	247
1.21	Compatibility with version 2.07 and below	252
2	Prefix Support (glossaries-prefix Code)	254
3	Glossary Styles	261
3.1	Glossary hyper-navigation definitions (glossary-hypernav package)	261
3.2	In-line Style (glossary-inline.sty)	263
3.3	List Style (glossary-list.sty)	265
3.4	Glossary Styles using longtable (the glossary-long package)	269
3.5	Glossary Styles using longtable and booktabs (the glossary-longbooktabs) package	275
3.6	Glossary Styles using longtable (the glossary-longragged package)	280
3.7	Glossary Styles using multicol (glossary-mcols.sty)	285
3.8	Glossary Styles using supertabular environment (glossary-super package)	291
3.9	Glossary Styles using supertabular environment (glossary-superragged package)	298
3.10	Tree Styles (glossary-tree.sty)	304

4 Backwards Compatibility	314
4.1 glossaries-compatible-207	314
4.2 glossaries-compatible-307	320
5 Accessibility Support (glossaries-accsupp Code)	334
5.1 Defining Replacement Text	335
5.2 Accessing Replacement Text	338
5.3 Displaying the Glossary	354
5.4 Acronyms	355
5.5 Debugging Commands	370
6 Multi-Lingual Support	372
6.1 Polyglossia Captions	372
Glossary	374
Change History	375
Index	398

1 Main Package Code

1.1 Package Definition

This package requires $\text{\LaTeX}2_{\epsilon}$.

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries}[2017/01/19 v4.29 (NLCT)]
```

Required packages:

```
3 \RequirePackage{ifthen}
4 \RequirePackage{xkeyval}[2006/11/18]
5 \RequirePackage{mfirstuc}
```

The textcase package has much better case changing handling, so use `\MakeTextUppercase` instead of `\MakeUppercase`

```
6 \RequirePackage{textcase}
7 \renewcommand*{\mfirstucMakeUppercase}{\MakeTextUppercase}%
8 \RequirePackage{xfor}
```

```
9 \RequirePackage{datatool-base}
```

Need to use `\new@ifnextchar` instead of `\@ifnextchar` in commands that have a final optional argument (such as `\gls`) so require `.` Thanks to Morten Høgholm for suggesting this. (This has replaced using the `xspace` package.)

```
10 \RequirePackage{amsgen}
```

As from v3.0, now loading `etoolbox`:

```
11 \RequirePackage{etoolbox}
```

Check if doc has been loaded.

```
f@gls@docloaded
```

```
12 \newif\if@gls@docloaded
13 \@ifpackageloaded{doc}%
14 {%
15   \@gls@docloadedtrue
16 }%
17 {%
18   \@ifclassloaded{nlctdoc}{\@gls@docloadedtrue}{\@gls@docloadedfalse}%
19 }
20 \if@gls@docloaded
```

\doc has been loaded, so some modifications need to be made to ensure both packages can work together. The amount of conflict has been reduced as from v4.11 and no longer involves patching internal commands.

\PrintChanges needs to use doc's version of theglossary, so save that.

org@theglossary

```
21 \let\glsorg@theglossary\theglossary
```

@endtheglossary

```
22 \let\glsorg@endtheglossary\endtheglossary
```

\PrintChanges Now redefine \PrintChanges so that it uses the original theglossary environment.

```
23 \let\glsorg@PrintChanges\PrintChanges
24 \renewcommand{\PrintChanges}{%
25   \begingroup
26     \let\theglossary\glsorg@theglossary
27     \let\endtheglossary\glsorg@endtheglossary
28     \glsorg@PrintChanges
29   \endgroup
30 }
```

End of doc stuff.

```
31 \fi
```

1.2 Package Options

debug Switch on debug mode. This will also cancel the nowarn option.

```
32 \define@boolkey{glossaries.sty}{@gls@}{debug}[true]{%
33   \if@gls@debug
34     \renewcommand*{\GlossariesWarning}[1]{%
35       \PackageWarning{glossaries}{##1}%
36     }%
37     \renewcommand*{\GlossariesWarningNoLine}[1]{%
38       \PackageWarningNoLine{glossaries}{##1}%
39     }%
40     \PackageInfo{glossaries}{debug mode ON (nowarn option disabled)}%
41   \else
42     \PackageInfo{glossaries}{debug mode OFF}%
43   \fi
44 }
```

Determine what to do if the see key is used before \makeglossaries. The default is to produce an error.

gls@see@noindex

```
45 \newcommand*{\@gls@see@noindex}{%
46   \PackageError{glossaries}{%

```

```

47 {'see' key may only be used after \string\makeglossaries\space
48 or \string\makenoidxglossaries}%
49 {You must use \string\makeglossaries\space
50 or \string\makenoidxglossaries\space before defining
51 any entries that have a 'see' key}%
52 }

```

seenoinindex

```

53 \define@choicekey{glossaries.sty}{seenoinindex}[\val\nr]{error,warn,ignore}{%
54   \ifcase\nr
55     \renewcommand*{\@gls@see@noindex}{%
56       \PackageError{glossaries}%
57       {'see' key may only be used after \string\makeglossaries\space
58       or \string\makenoidxglossaries}%
59       {You must use \string\makeglossaries\space
60       or \string\makenoidxglossaries\space before defining
61       any entries that have a 'see' key}%
62     }%
63   \or
64     \renewcommand*{\@gls@see@noindex}{%
65       \GlossariesWarning{'see' key ignored}%
66     }%
67   \or
68     \renewcommand*{\@gls@see@noindex}{}%
69   \fi
70 }

```

toc The toc package option will add the glossaries to the table of contents. This is a boolean key, if the value is omitted it is taken to be true.

```
71 \define@boolkey{glossaries.sty}[gls]{toc}[true]{}
```

numberline The numberline package option adds \numberline to \addcontentsline. Note that this option only has an effect if used in with toc=true.

```
72 \define@boolkey{glossaries.sty}[gls]{numberline}[true]{}
```

\@@glossarysec The sectional unit used to start the glossary is stored in \@@glossarysec. If chapters are defined, this is initialised to chapter, otherwise it is initialised to section.

```

73 \ifcsundef{chapter}%
74   {\newcommand*{\@@glossarysec}{section}}%
75   {\newcommand*{\@@glossarysec}{chapter}}

```

section The section key can be used to set the sectional unit. If no unit is specified, use section as the default. The starred form of the named sectional unit will be used. If you want some other way to start the glossary section (e.g. a numbered section) you will have to redefined \glossarysection.

```

76 \define@choicekey{glossaries.sty}{section}{part,chapter,section,%
77 subsection,subsubsection,paragraph,subparagraph}[section]{%
78   \renewcommand*{\@@glossarysec}{#1}}

```

Determine whether or not to use numbered sections.

glossarysecstar

```
79 \newcommand*{\@@glossarysecstar}{*}
```

glossaryseclabel

```
80 \newcommand*{\@@glossaryseclabel}{}
```

\glsautoprefix

Prefix to add before label if automatically generated:

```
81 \newcommand*{\glsautoprefix}{}
```

numberedsection

```
82 \define@choicekey{glossaries.sty}{numberedsection}[\val\nr]{%
83 false,nolabel,autolabel,nameref}[nolabel]{%
84   \ifcase\nr\relax
85     \renewcommand*{\@@glossarysecstar}{*}%
86     \renewcommand*{\@@glossaryseclabel}{}%
87   \or
88     \renewcommand*{\@@glossarysecstar}{}%
89     \renewcommand*{\@@glossaryseclabel}{}%
90   \or
91     \renewcommand*{\@@glossarysecstar}{}%
92     \renewcommand*{\@@glossaryseclabel}{%
93       \label{\glsautoprefix\@glo@type}}%
94   \or
95     \renewcommand*{\@@glossarysecstar}{*}%
96     \renewcommand*{\@@glossaryseclabel}{%
97       \protected@edef\@currentlabelname{\glossarytoctitle}%
98       \label{\glsautoprefix\@glo@type}}%
99   \fi
100 }
```

The default glossary style is stored in `\@glossary@default@style`. This is initialised to `list`. (The `list` style is defined in the accompanying package described in [section 1.19](#).) Note that the `list` style is incompatible with `classicthesis` so change the default to `index` if that package has been loaded.

y@default@style

```
101 \ifpackageloaded{classicthesis}
102 {\newcommand*{\@glossary@default@style}{index}}
103 {\newcommand*{\@glossary@default@style}{list}}
```

style

The default glossary style can be changed using the `style` package option. The value can be the name of any defined glossary style. The glossary style is set at the beginning of the document, so you can still use the `style` key to set a style that is defined in another package. This package comes with some predefined styles that are defined in [section 1.19](#).

```
104 \define@key{glossaries.sty}{style}{%
105   \renewcommand*{\@glossary@default@style}{#1}%
106 }
```

Each `\DeclareOptionX` needs a corresponding `\DeclareOption` so that it can be passed as a document class option, so define a command that will implement both.

`s@declareoption`

```
107 \newcommand*{\@gls@declareoption}[2]{%
108   \DeclareOptionX{#1}{#2}%
109   \DeclareOption{#1}{#2}%
110 }
```

Each entry within a given glossary will have an associated number list. By default, this refers to the page numbers on which that entry has been used, but it can also refer to any counter used in the document (such as the section or equation counters). The default number list format displays the number list “as is”:

`aryentrynumbers`

```
111 \newcommand*{\glossaryentrynumbers}[1]{#1\gls@save@numberlist{#1}}
```

`nonumberlist` Note that the entire number list for a given entry will be passed to `\glossaryentrynumbers` so any font changes will also be applied to the delimiters. The `nonumberlist` package option suppresses the number lists (this simply redefines `\glossaryentrynumbers` to ignore its argument).

```
112 \@gls@declareoption{nonumberlist}{%
113   \renewcommand*{\glossaryentrynumbers}[1]{\gls@save@numberlist{#1}}%
114 }
```

`savenumberlist` Provide means to store the number list for entries.

```
115 \define@boolkey{glossaries.sty}{gls}{savenumberlist}[true]{%
116   \glssavenumberlistfalse}
```

`eautionumberlist`

```
117 \newcommand*{\@glo@seeautonumberlist{}}
```

`eautionumberlist` Automatically activates number list for entries containing the see key.

```
118 \@gls@declareoption{seeautonumberlist}{%
119   \renewcommand*{\@glo@seeautonumberlist}{%
120     \def\@glo@prefix{\glsnextpages}%
121     }%
122 }
```

`\@gls@loadlong`

```
123 \newcommand*{\@gls@loadlong}{\RequirePackage{glossary-long}}
```

`nolong` This option prevents from being loaded. This means that the glossary styles that use the longtable environment will not be available. This option is provided to reduce overhead caused by loading unrequired packages.

```
124 \@gls@declareoption{nolong}{\renewcommand*{\@gls@loadlong}{}}
```


`\@gls@loadsuper` The package isn't loaded if isn't installed.

```

125 \IfFileExists{supertabular.sty}{%
126   \newcommand*{\@gls@loadsuper}{\RequirePackage{glossary-super}}}%
127   \newcommand*{\@gls@loadsuper}{}

```

`nosuper` This option prevents from being loaded. This means that the glossary styles that use the supertabular environment will not be available. This option is provided to reduce overhead caused by loading unrequired packages.

```

128 \@gls@declareoption{nosuper}{\renewcommand*{\@gls@loadsuper}{}

```

`\@gls@loadlist`

```

129 \newcommand*{\@gls@loadlist}{\RequirePackage{glossary-list}}

```

`nolist` This option prevents from being loaded (to reduce overheads if required). Naturally, the styles defined in will not be available if this option is used.

```

130 \@gls@declareoption{nolist}{\renewcommand*{\@gls@loadlist}{}

```

`\@gls@loadtree`

```

131 \newcommand*{\@gls@loadtree}{\RequirePackage{glossary-tree}}

```

`notree` This option prevents from being loaded (to reduce overheads if required). Naturally, the styles defined in will not be available if this option is used.

```

132 \@gls@declareoption{notree}{\renewcommand*{\@gls@loadtree}{}

```

`nostyles` Provide an option to suppress all the predefined styles (in the event that the user has custom styles that are not dependent on the predefined styles).

```

133 \@gls@declareoption{nostyles}{%
134   \renewcommand*{\@gls@loadlong}{}%
135   \renewcommand*{\@gls@loadsuper}{}%
136   \renewcommand*{\@gls@loadlist}{}%
137   \renewcommand*{\@gls@loadtree}{}%
138   \let\@glossary@default@style\relax
139 }

```

`postdescription` The description terminator is given by `\glspostdescription` (except for the 3 and 4 column styles). This is a full stop by default. The spacefactor is adjusted in case the description ends with an upper case letter. (Patch provided by Michael Pock.)

```

140 \newcommand*{\glspostdescription}{%
141   \ifglsnopostdot\else.\spacefactor\sfcode'\. \fi
142 }

```

`nopostdot` Boolean option to suppress post description dot

```

143 \define@boolkey{glossaries.sty}[gls]{nopostdot}[true]{}
144 \glsnopostdotfalse

```

`nogroupskip` Boolean option to suppress vertical space between groups in the pre-defined styles.

```

145 \define@boolkey{glossaries.sty}[gls]{nogroupskip}[true]{}
146 \glsnogroupskipfalse

```

ucmark Boolean option to determine whether or not to use upper case in definition of \glsglossarymark

```
147\define@boolkey{glossaries.sty}[gls]{ucmark}[true]{}
148\@ifclassloaded{memoir}
149{%
150  \glsucmarktrue
151}%
152{%
153  \glsucmarkfalse
154}
```

entrycounter Defines a counter that can be used in the standard glossary styles to number each (main) entry. If true, this will define a counter called glossaryentry.

```
155\define@boolkey{glossaries.sty}[gls]{entrycounter}[true]{}
156\glsentrycounterfalse
```

entrycounterwithin This option can be used to set a parent counter for glossaryentry. This option automatically sets entrycounter=true.

```
157\define@key{glossaries.sty}{counterwithin}{%
158  \renewcommand*{\@gls@counterwithin}{#1}%
159  \glsentrycountertrue
160}
```

s@counterwithin The default value is no parent counter:

```
161\newcommand*{\@gls@counterwithin}{}%
```

subentrycounter Define a counter that can be used in the standard glossary styles to number each level 1 entry. If true, this will define a counter called glossarysubentry.

```
162\define@boolkey{glossaries.sty}[gls]{subentrycounter}[true]{}
163\glssubentrycounterfalse
```

default@sorttype Initialise default sort for \printnoidxglossary

```
164\newcommand*{\@gls@default@sorttype}{standard}
```

sort Define the sort method: sort=standard (default), sort=def (order of definition) or sort=use (order of use).

```
165\define@choicekey{glossaries.sty}{sort}{standard,def,use}{%
166  \renewcommand*{\@gls@default@sorttype}{#1}%
167  \csname @gls@setupsort@#1\endcsname
168}
```

sprestandardsort

```
\glsprestandardsort{<sort cs>}{<type>}{<label>}
```

Allow user to hook into sort mechanism. The first argument *<sort cs>* is the temporary control sequence containing the sort value before it has been sanitized and had `makeindex/xindy` special characters escaped.

```

169 \newcommand*{\glsprestandardsort}[3]{%
170   \glsdosanitizesort
171 }

upsort@standard  Set up the macros for default sorting.
172 \newcommand*{\@gls@setupsort@standard}{%
  Store entry information when it's defined.
173   \def\do@glo@storeentry{\@glo@storeentry}%
  No count register required for standard sort.
174   \def\@gls@defsortcount##1{%
    Sort according to sort key (\@glo@sort) if provided otherwise sort according to the entry's
    name (\@glo@name). (First argument glossary type, second argument entry label.)
175   \def\@gls@defsort##1##2{%
176     \ifx\@glo@sort\@glsdefaultsort
177       \let\@glo@sort\@glo@name
178     \fi
179     \let\glsdosanitizesort\@gls@sanitizesort
180     \glsprestandardsort{\@glo@sort}{##1}{##2}%
181     \expandafter\protected@xdef\csname glo@##2@sort\endcsname{\@glo@sort}%
182   }%

  Don't need to do anything when the entry is used.
183   \def\@gls@setsort##1{%
184   }

  Set standard sort as the default:
185 \@gls@setupsort@standard

lssortnumberfmt  Format the number used as the sort key by sort=def and sort=use. Defaults to six digit num-
bering.
186 \newcommand*{\glssortnumberfmt}[1]{%
187   \ifnum#1<100000 0\fi
188   \ifnum#1<10000 0\fi
189   \ifnum#1<1000 0\fi
190   \ifnum#1<100 0\fi
191   \ifnum#1<10 0\fi
192   \number#1%
193 }

s@setupsort@def  Set up the macros for order of definition sorting.
194 \newcommand*{\@gls@setupsort@def}{%
  Store entry information when it's defined.
195   \def\do@glo@storeentry{\@glo@storeentry}%

```

Defined count register associated with the glossary.

```
196 \def\@gls@defsortcount##1{%
197   \expandafter\global
198   \expandafter\newcount\csname glossary@##1@sortcount\endcsname
199 }%
```

Increment count register associated with the glossary and use as the sort key.

```
200 \def\@gls@defsort##1##2{%
201   \expandafter\global\expandafter
202   \advance\csname glossary@##1@sortcount\endcsname by 1\relax
203   \expandafter\protected\edef\csname glo@##2@sort\endcsname{%
204     \expandafter\glssortnumberfmt
205     {\csname glossary@##1@sortcount\endcsname}}%
206 }%
```

Don't need to do anything when the entry is used.

```
207 \def\@gls@setsort##1{%
208 }
```

s@setupsort@use Set up the macros for order of use sorting.

```
209 \newcommand*{\@gls@setupsort@use}{%
```

Don't store entry information when it's defined.

```
210 \let\do@glo@storeentry\@gobble
```

Defined count register associated with the glossary.

```
211 \def\@gls@defsortcount##1{%
212   \expandafter\global
213   \expandafter\newcount\csname glossary@##1@sortcount\endcsname
214 }%
```

Initialise the sort key to empty.

```
215 \def\@gls@defsort##1##2{%
216   \expandafter\gdef\csname glo@##2@sort\endcsname{%
217 }%
```

If the sort key hasn't been set, increment the counter associated with the glossary and set the sort key.

```
218 \def\@gls@setsort##1{%
```

Get the parent, if one exists

```
219   \edef\@glo@parent{\csname glo@##1@parent\endcsname}%
```

Set the information for the parent entry if not already done.

```
220   \ifx\@glo@parent\@empty
221   \else
222     \expandafter\@gls@setsort\expandafter{\@glo@parent}%
223   \fi
```

Set index information for this entry

```
224   \edef\@glo@type{\csname glo@##1@type\endcsname}%
225   \edef\@gls@tmp{\csname glo@##1@sort\endcsname}%
```

```

226 \ifx\@gls@tmp\@empty
227 \expandafter\global\expandafter
228 \advance\csname glossary@\@glo@type @sortcount\endcsname by 1\relax
229 \expandafter\protected\xdef\csname glo@##1@sort\endcsname{%
230 \expandafter\glssortnumberfmt
231 {\csname glossary@\@glo@type @sortcount\endcsname}}%
232 \@glo@storeentry{##1}%
233 \fi
234 }%
235 }

```

`\glsdefmain` Define the main glossary. This will be the first glossary to be displayed when using `\printglossaries`. The default extensions conflict if used with `doc`, so provide different extensions if `doc` loaded. (If these extensions are inappropriate, use `nomain` and manually define the main glossary with the desired extensions.)

```

236 \newcommand*{\glsdefmain}{%
237 \if@gls@docloaded
238 \newglossary[glg2]{main}{gls2}{glo2}{\glossaryname}%
239 \else
240 \newglossary{main}{gls}{glo}{\glossaryname}%
241 \fi

```

Define hook to set the toc title when translator is in use.

```

242 \newcommand*{\gls@tr@set@main@toctitle}{%
243 \translatelet{\glossarytoctitle}{Glossary}%
244 }%
245 }

```

Keep track of the default glossary. This is initialised to the main glossary, but can be changed if for some reason you want to make a secondary glossary the main glossary. This affects any commands that can optionally take a glossary name as an argument (or as the value of the type key in a key-value list). This was mainly done so that `\loadglsentries` can temporarily change `\glsdefaulttype` while it loads a file containing new glossary entries (see [section 1.10](#)).

`\glsdefaulttype`

```

246 \newcommand*{\glsdefaulttype}{main}

```

Keep track of which glossary the acronyms are in. This is initialised to `\glsdefaulttype`, but is changed by the acronym package option.

`\acronymtype`

```

247 \newcommand*{\acronymtype}{\glsdefaulttype}

```

`nomain` The `nomain` option suppress the creation of the main glossary.

```

248 \@gls@declareoption{nomain}{%
249 \let\glsdefaulttype\relax
250 \renewcommand*{\glsdefmain}{}%
251 }

```

acronym The acronym option sets an associated conditional which is used in [section 1.17](#) to determine whether or not to define a separate glossary for acronyms.

```
252 \define@boolkey{glossaries.sty}{gls}{acronym}[true]{%
253   \ifglsacronym
254     \renewcommand{\@gls@do@acronymsdef}{%
255       \DeclareAcronymList{acronym}%
256       \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
257       \renewcommand*\@acronymtype{acronym}%
```

Define hook to set the toc title when translator is in use.

```
258     \newcommand*\@gls@tr@set@acronym@toctitle{%
259       \translatelet{\glossarytoctitle}{Acronyms}%
260     }%
261   }%
262 \else
263   \let\@gls@do@acronymsdef\relax
264 \fi
265 }
```

\printacronyms Define \printacronyms at the start of the document if acronym is set and compatibility mode isn't on and \printacronyms hasn't already been defined.

```
266 \AtBeginDocument{%
267   \ifglsacronym
268     \ifbool{glscompatible-3.07}%
269       {}%
270     {%
271       \providecommand*\printacronyms[1][]{%
272         \printglossary[type=\acronymtype,#1]}%
273     }%
274 \fi
275 }
```

@do@acronymsdef Set default value

```
276 \newcommand*\@gls@do@acronymsdef{}
```

acronyms Provide a synonym for acronym=true that can be passed via the document class options.

```
277 \@gls@declareoption{acronyms}{%
278   \glsacronymtrue
279   \renewcommand{\@gls@do@acronymsdef}{%
280     \DeclareAcronymList{acronym}%
281     \newglossary[alg]{acronym}{acr}{acn}{\acronymname}%
282     \renewcommand*\@acronymtype{acronym}%
```

Define hook to set the toc title when translator is in use.

```
283     \newcommand*\@gls@tr@set@acronym@toctitle{%
284       \translatelet{\glossarytoctitle}{Acronyms}%
285     }%
286   }%
287 }
```

`glsacronymlists` Comma-separated list of glossary labels indicating which glossaries contain acronyms. Note that `\SetAcronymStyle` must be used after adding labels to this macro.

```
288 \newcommand*{\@glsacronymlists}{}
```

`addtoacronymlists`

```
289 \newcommand*{\@addtoacronymlists}[1]{%
290   \ifx\@glsacronymlists\@empty
291     \protected@xdef\@glsacronymlists{#1}%
292   \else
293     \protected@xdef\@glsacronymlists{\@glsacronymlists,#1}%
294   \fi
295 }
```

`declareAcronymList` Identifies the named glossary as a list of acronyms and adds to the list. (Doesn't check if the glossary exists, but checks if label already in list. Use `\SetAcronymStyle` after identifying all the acronym lists.)

```
296 \newcommand*{\DeclareAcronymList}[1]{%
297   \glsIfListOfAcronyms{#1}{\@addtoacronymlists{#1}}%
298 }
```

`IfListOfAcronyms`

```
\glsIfListOfAcronyms{<label>}{<true part>}{<false part>}
```

Determines if the glossary with the given label has been identified as being a list of acronyms.

```
299 \newcommand{\glsIfListOfAcronyms}[1]{%
300   \edef\@do@gls@islistofacronyms{%
301     \noexpand\@gls@islistofacronyms{#1}{\@glsacronymlists}}%
302   \@do@gls@islistofacronyms
303 }
```

Internal command requires label and list to be expanded:

```
304 \newcommand{\@gls@islistofacronyms}[4]{%
305   \def\gls@islistofacronyms##1,#1,##2\end@gls@islistofacronyms{%
306     \def\@before{##1}\def\@after{##2}}%
307   \gls@islistofacronyms,#2,#1,\@nil\end@gls@islistofacronyms
308   \ifx\@after\@nnil
```

Not found

```
309   #4%
310   \else
```

Found

```
311   #3%
312   \fi
313 }
```

`lsglsacronymlist` Convenient boolean.

```
314 \newif\if@lsglsacronymlist
```

`ckisacronymlist` Sets the above boolean if argument is a label representing a list of acronyms.

```
315 \newcommand*{\gls@ckisacronymlist}[1]{%
316   \glsIfListOfAcronyms{#1}%
317   {\@glsisacronymlisttrue}{\@glsisacronymlistfalse}%
318 }
```

`SetAcronymLists` Sets the “list of acronyms” list. Argument must be a comma-separated list of glossary labels. (Doesn’t check at this point if the glossaries exists.)

```
319 \newcommand*{\SetAcronymLists}[1]{%
320   \renewcommand*{\@glsacronymlists}{#1}%
321 }
```

`acronymlists`

```
322 \define@key{glossaries.sty}{acronymlists}{%
323   \DeclareAcronymList{#1}%
324 }
```

The default counter associated with the numbers in the glossary is stored in `\glscounter`. This is initialised to the page counter. This is used as the default counter when a new glossary is defined, unless a different counter is specified in the optional argument to `\newglossary` (see [section 1.6](#)).

`\glscounter`

```
325 \newcommand{\glscounter}{page}
```

`counter` The counter option changes the default counter. (This just redefines `\glscounter`.)

```
326 \define@key{glossaries.sty}{counter}{%
327   \renewcommand*{\glscounter}{#1}%
328 }
```

`gls@nohyperlist`

```
329 \newcommand*{\@gls@nohyperlist}{}%
```

`lareNoHyperList`

```
330 \newcommand*{\GlsDeclareNoHyperList}[1]{%
331   \ifdefempty\@gls@nohyperlist
332   {%
333     \renewcommand*{\@gls@nohyperlist}{#1}%
334   }%
335   {%
336     \appto\@gls@nohyperlist{,#1}%
337   }%
338 }
```

`nohypertypes`

```
339 \define@key{glossaries.sty}{nohypertypes}{%
340   \GlsDeclareNoHyperList{#1}%
341 }
```


`glossariesWarning` Prints a warning message.

```
342 \newcommand*{\GlossariesWarning}[1]{%
343   \PackageWarning{glossaries}{#1}%
344 }
```

`glossariesWarningNoLine` Prints a warning message without the line number.

```
345 \newcommand*{\GlossariesWarningNoLine}[1]{%
346   \PackageWarningNoLine{glossaries}{#1}%
347 }
```

`nowarn` Define package option to suppress warnings

```
348 \@gls@declareoption{nowarn}{%
349   \if@gls@debug
350     \GlossariesWarning{Warnings can't be suppressed in debug mode}%
351   \else
352     \renewcommand*{\GlossariesWarning}[1]{}%
353     \renewcommand*{\GlossariesWarningNoLine}[1]{}%
354   \fi
355 }
```

`warnonglossdefined` Issue a warning if overriding `\printglossary`

```
356 \newcommand*{\@gls@warnonglossdefined}{%
357   \GlossariesWarning{Overriding \string\printglossary}%
358 }
```

`warntheglossdefined` Issue a warning if overriding `theglossary`

```
359 \newcommand*{\@gls@warnontheglossdefined}{%
360   \GlossariesWarning{Overriding 'theglossary' environment}%
361 }
```

`noundefwarn` Suppress warning on redefinition of `\printglossary`

```
362 \@gls@declareoption{noundefwarn}{%
363   \renewcommand*{\@gls@warnonglossdefined}{}%
364   \renewcommand*{\@gls@warnontheglossdefined}{}%
365 }
```

As from version 3.08a, the only information written to the external glossary files are the label and sort values. Therefore, now, the only `sanitize` option that makes sense is the one for the sort key. so the `sanitize` option is now deprecated and there is only a `sanitizesort` option.

`sanitizedesc`

```
366 \newcommand*{\@gls@sanitizedesc}{%
367 }
```

`setexpandfield` `\glssetexpandfield{<field>}`

Sets field to always expand.

```

368 \newcommand*{\glsetexpandfield}[1]{%
369   \csdef{gls@assign@#1@field}##1##2{%
370     \@@gls@expand@field{##1}{#1}{##2}%
371   }%
372 }

```

setnoexpandfield `\glsetnoexpandfield{<field>}`

Sets field to never expand.

```

373 \newcommand*{\glsetnoexpandfield}[1]{%
374   \csdef{gls@assign@#1@field}##1##2{%
375     \@@gls@noexpand@field{##1}{#1}{##2}%
376   }%
377 }

```

sign@type@field The type must always be expandable.

```
378 \glsetexpandfield{type}
```

sign@desc@field The description is not expanded by default:

```
379 \glsetnoexpandfield{desc}
```

escplural@field

```
380 \glsetnoexpandfield{descplural}
```

ls@sanitizename

```
381 \newcommand*{\@gls@sanitizename}{}
```

sign@name@field Don't expand name by default.

```
382 \glsetnoexpandfield{name}
```

@sanitizesymbol

```
383 \newcommand*{\@gls@sanitizesymbol}{}
```

gn@symbol@field Don't expand symbol by default.

```
384 \glsetnoexpandfield{symbol}
```

bolplural@field

```
385 \glsetnoexpandfield{symbolplural}
```

Sanitizing stuff:

ls@sanitizesort

```

386 \newcommand*{\@gls@sanitizesort}{%
387   \ifglssanitizesort
388     \@@gls@sanitizesort
389   \else
390     \@@gls@nosanitizesort
391   \fi
392 }

```

ls@sanitizesort

```
393 \newcommand*\@@gls@sanitizesort{%
394   \@onelevel@sanitize\@glo@sort
395 }
```

@nosanitizesort

```
396 \newcommand*\@@gls@nosanitizesort{}
```

dx@sanitizesort Remove braces around first character (if present) before sanitizing.

```
397 \newcommand*\@gls@noidx@sanitizesort{%
398   \ifdefvoid\@glo@sort
399   {}%
400   {%
401     \expandafter\@@gls@noidx@sanitizesort\@glo@sort\gls@end@sanitizesort
402   }%
403 }
404 \def\@@gls@noidx@sanitizesort#1#2\gls@end@sanitizesort{%
405   \def\@glo@sort{#1#2}%
406   \@onelevel@sanitize\@glo@sort
407 }
```

@nosanitizesort

```
408 \newcommand*\@@gls@noidx@nosanitizesort{%
409   \ifdefvoid\@glo@sort
410   {}%
411   {%
412     \expandafter\@@gls@noidx@no@sanitizesort\@glo@sort\gls@end@sanitizesort
413   }%
414 }
415 \def\@@gls@noidx@no@sanitizesort#1#2\gls@end@sanitizesort{%
416   \bgroup
417   \glsnoidxstripaccents
418   \protected@xdef\@glo@sort{#1#2}%
419   \egroup
420   \let\@glo@sort\@glo@sort
421 }
```

idxstripaccents

```
422 \newcommand*\glsnoidxstripaccents{%
423   \let\IeC\@firstofone
424   \let\' \@firstofone
425   \let\' \@firstofone
426   \let\~ \@firstofone
427   \let\" \@firstofone
428   \let\u \@firstofone
429   \let\t \@firstofone
430   \let\d \@firstofone
431   \let\r \@firstofone
432   \let\= \@firstofone
```

```

433 \let\.\@firstofone
434 \let\~\@firstofone
435 \let\v\@firstofone
436 \let\H\@firstofone
437 \let\c\@firstofone
438 \let\b\@firstofone
439 \def\AE{AE}%
440 \def\ae{ae}%
441 \def\OE{OE}%
442 \def\oe{oe}%
443 \def\AA{AA}%
444 \def\aa{aa}%
445 \def\L{L}%
446 \def\l{l}%
447 \def\O{O}%
448 \def\o{o}%
449 \def\SS{SS}%
450 \def\ss{ss}%
451 \def\th{th}%
452 }

```

Before defining the sanitize package option, The key-value list for the sanitize value needs to be defined. These are all boolean keys. If they are not given a value, assume true.

```

453 \define@boolkey[glS]{sanitize}{description}[true]{%
454   \GlossariesWarning{sanitize={description} package option deprecated}%
455   \ifglS@sanitize@description
456     \glSsetnoexpandfield{desc}%
457     \glSsetnoexpandfield{descplural}%
458   \else
459     \glSsetexpandfield{desc}%
460     \glSsetexpandfield{descplural}%
461   \fi
462 }

463 \define@boolkey[glS]{sanitize}{name}[true]{%
464   \GlossariesWarning{sanitize={name} package option deprecated}%
465   \ifglS@sanitize@name
466     \glSsetnoexpandfield{name}%
467   \else
468     \glSsetexpandfield{name}%
469   \fi
470 }

471 \define@boolkey[glS]{sanitize}{symbol}[true]{%
472   \GlossariesWarning{sanitize={symbol} package option deprecated}%
473   \ifglS@sanitize@symbol
474     \glSsetnoexpandfield{symbol}%
475     \glSsetnoexpandfield{symbolplural}%
476   \else
477     \glSsetexpandfield{symbol}%

```

```

478 \glsssetexpandfield{symbolplural}%
479 \fi
480 }

```

sanitizesort

```

481 \define@boolkey{glossaries.sty}[gls]{sanitizesort}[true]{%
482 \ifglsssanitizesort
483 \glsssetnoexpandfield{sortvalue}%
484 \renewcommand*{\@gls@noidx@setsanitizesort}{%
485 \glsssanitizesorttrue
486 \glsssetnoexpandfield{sortvalue}%
487 }%
488 \else
489 \glsssetexpandfield{sortvalue}%
490 \renewcommand*{\@gls@noidx@setsanitizesort}{%
491 \glsssanitizesortfalse
492 \glsssetexpandfield{sortvalue}%
493 }%
494 \fi
495 }

```

Default setting:

```

496 \glsssanitizesorttrue
497 \glsssetnoexpandfield{sortvalue}%

```

setsanitizesort Default behaviour for \makenoidxglossaries is sanitizesort=false.

```

498 \newcommand*{\@gls@noidx@setsanitizesort}{%
499 \glsssanitizesortfalse
500 \glsssetexpandfield{sortvalue}%
501 }

502 \define@choicekey[gls]{sanitize}{sort}{true,false}[true]{%
503 \setbool{glsssanitizesort}{#1}%
504 \ifglsssanitizesort
505 \glsssetnoexpandfield{sortvalue}%
506 \else
507 \glsssetexpandfield{sortvalue}%
508 \fi
509 \GlossariesWarning{sanitize={sort} package option
510 deprecated. Use sanitizesort instead}%
511 }

```

sanitize

```

512 \define@key{glossaries.sty}{sanitize}[description=true,symbol=true,name=true]{%
513 \ifthenelse{\equal{#1}{none}}{%
514 {%
515 \GlossariesWarning{sanitize package option deprecated}%
516 \glsssetexpandfield{name}%
517 \glsssetexpandfield{symbol}%
518 \glsssetexpandfield{symbolplural}%

```

```

519     \glssetexpandfield{desc}%
520     \glssetexpandfield{descplural}%
521 }%
522 {%
523     \setkeys[gls]{sanitize}{#1}%
524 }%
525 }

\ifglstranslate As from version 3.13a, the translator package option is a choice rather than boolean option
                so now need to define conditional:
526 \newif\ifglstranslate

notranslatorhook \@gls@notranslatorhook has been removed.

s@usetranslator
527 \newcommand*\@gls@usetranslator{%
    polyglossia tricks \@ifpackageloaded into thinking that babel has been loaded, so check for
    polyglossia as well.
528     \@ifpackageloaded{polyglossia}%
529     {%
530         \let\glsifusetranslator\@secondoftwo
531     }%
532     {%
533         \@ifpackageloaded{babel}%
534         {%
535             \IfFileExists{translator.sty}%
536             {%
537                 \RequirePackage{translator}%
538                 \let\glsifusetranslator\@firstoftwo
539             }%
540             {}%
541         }%
542     }%
543 }%
544 }

dtranslatordict Checks if given translator dictionary has been loaded.
545 \newcommand{\glsifusedtranslatordict}[3]{%
546     \glsifusetranslator
547     {\ifcsdef{ver@glossaries-dictionary-#1.dict}{#2}{#3}}%
548     {#3}%
549 }

notranslate Provide a synonym for translate=false that can be passed via the document class.
550 \@gls@declareoption{notranslate}{%
551     \glstranslatefalse
552     \let\@gls@usetranslator\relax
553     \let\glsifusetranslator\@secondoftwo
554 }

```

translate Define translate option. If false don't set up multi-lingual support.

```
555 \define@choicekey{glossaries.sty}{translate}[\val\nr]%
556 {true,false,babel}[true]%
557 {%
558   \ifcase\nr\relax
559     \glstranslatetrue
560     \renewcommand*\@gls@usetranslator{%
561       \@ifpackageloaded{polyglossia}%
562       {%
563         \let\glsifusetranslator\@secondoftwo
564       }%
565       {%
566         \@ifpackageloaded{babel}%
567         {%
568           \IfFileExists{translator.sty}%
569           {%
570             \RequirePackage{translator}%
571             \let\glsifusetranslator\@firstoftwo
572           }%
573         }%
574       }%
575     }%
576   }%
577 }%
578 \or
579   \glstranslatefalse
580   \let\@gls@usetranslator\relax
581   \let\glsifusetranslator\@secondoftwo
582 \or
583   \glstranslatetrue
584   \let\@gls@usetranslator\relax
585   \let\glsifusetranslator\@secondoftwo
586 \fi
587 }
```

Set the default value:

```
588 \glstranslatefalse
589 \let\glsifusetranslator\@secondoftwo
590 \@ifpackageloaded{translator}%
591 {%
592   \glstranslatetrue
593   \let\glsifusetranslator\@firstoftwo
594 }%
595 {%
596   \@for\gls@thissty:=tracklang,babel,ngerman,polyglossia\do
597   {
598     \@ifpackageloaded{\gls@thissty}%
599     {%
600       \glstranslatetrue
```

```

601     \@endfortrue
602   }%
603   {}%
604 }
605 }

```

indexonlyfirst Set whether to only index on first use.

```

606 \define@boolkey{glossaries.sty}[gls]{indexonlyfirst}[true]{}
607 \glsindexonlyfirstfalse

```

hyperfirst Set whether or not terms should have a hyperlink on first use.

```

608 \define@boolkey{glossaries.sty}[gls]{hyperfirst}[true]{}
609 \glshyperfirsttrue

```

gls@setacrstyle Keep track of whether an acronym style has been set (for the benefit of `\setupglossaries`):

```

610 \newcommand*{\@gls@setacrstyle}{}

```

footnote Set the long form of the acronym in footnote on first use.

```

611 \define@boolkey{glossaries.sty}[glsacr]{footnote}[true]{}%
612 \ifbool{glsacrdescription}%
613   {}%
614   {%
615     \renewcommand*{\@gls@sanitizedesc}{}%
616   }%
617 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
618 }

```

description Allow acronyms to have a description (needs to be set using the description key in the optional argument of `\newacronym`).

```

619 \define@boolkey{glossaries.sty}[glsacr]{description}[true]{}%
620 \renewcommand*{\@gls@sanitizesymbol}{}%
621 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
622 }

```

smallcaps Define `\newacronym` to set the short form in small capitals.

```

623 \define@boolkey{glossaries.sty}[glsacr]{smallcaps}[true]{}%
624 \renewcommand*{\@gls@sanitizesymbol}{}%
625 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
626 }

```

smaller Define `\newacronym` to set the short form using `\smaller` which obviously needs to be defined by loading the appropriate package.

```

627 \define@boolkey{glossaries.sty}[glsacr]{smaller}[true]{}%
628 \renewcommand*{\@gls@sanitizesymbol}{}%
629 \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
630 }

```


dua Define `\newacronym` to always use the long forms (i.e. don't use acronyms)

```

631 \define@boolkey{glossaries.sty}[glsacr]{dua}[true]{%
632   \renewcommand*{\@gls@sanitizesymbol}{}%
633   \renewcommand*{\@gls@setacrstyle}{\SetAcronymStyle}%
634 }

```

shortcuts Define acronym shortcuts.

```

635 \define@boolkey{glossaries.sty}[glsacr]{shortcuts}[true]{%

```

\glsorder Stores the glossary ordering. This may either be “word” or “letter”. This passes the relevant information to `makeglossaries`. The default is word ordering.

```

636 \newcommand*{\glsorder}{word}

```

\@glsorder The ordering information is written to the auxiliary file for `makeglossaries`, so ignore the auxiliary information.

```

637 \newcommand*{\@glsorder}[1]{}

```

order

```

638 \define@choicekey{glossaries.sty}{order}{word,letter}{%
639   \def\glsorder{#1}}

```

\ifglxindy Provide boolean to determine whether `xindy` or `makeindex` will be used to sort the glossaries.

```

640 \newif\ifglxindy

```

The default is `makeindex`:

```

641 \glxindyfalse

```

makeindex Define package option to specify that `makeindex` will be used to sort the glossaries:

```

642 \@gls@declareoption{makeindex}{\glxindyfalse}

```

The `xindy` package option may have a value which in turn can be a key=value list. First define the keys for this sub-list. The boolean `glsnumbers` determines whether to automatically add the `glsnumbers` letter group.

```

643 \define@boolkey[gl]{xindy}{glsnumbers}[true]{%
644 \gls@xindy@glsnumberstrue

```

y@main@language Define what language to use for each glossary type (if a language is not defined for a particular glossary type the language specified for the main glossary is used.)

```

645 \def\@xdy@main@language{\language}%

```

Define key to set the language

```

646 \define@key[gl]{xindy}{language}{\def\@xdy@main@language{#1}}

```

`\gls@codepage` Define the code page. If `\inputencodingname` is defined use that, otherwise have initialise with no codepage.

```

647 \ifcsundef{inputencodingname}{%
648   \def\gls@codepage{}}{%
649   \def\gls@codepage{\inputencodingname}
650 }

```

Define a key to set the code page.

```

651 \define@key[gls]{xindy}{codepage}{\def\gls@codepage{#1}}

```

`xindy` Define package option to specify that xindy will be used to sort the glossaries:

```

652 \define@key{glossaries.sty}{xindy}[]{%
653   \glsxindytrue
654   \setkeys[gls]{xindy}{#1}%
655 }

```

`xindygloss` Provide a synonym for xindy that can be passed via the document class options.

```

656 \@gls@declareoption{xindygloss}{%
657   \glsxindytrue
658 }

```

`xindynoglsnumbers` Provide a synonym for `xindy=glsnumbers=false` that can be passed via the document class options.

```

659 \@gls@declareoption{xindynoglsnumbers}{%
660   \glsxindytrue
661   \gls{xindy=glsnumbersfalse}
662 }

```

`automake` If this setting is on, automatically run `makeindex/xindy` at the end of the document. Must be used with `\makeglossaries`. Default is false.

```

663 \define@boolkey{glossaries.sty}[gls]{automake}[true]{%
664   \ifglsautomake
665     \renewcommand*{\@gls@doautomake}{%
666       \PackageError{glossaries}{You must use
667         \string\makeglossaries\space with automake=true}
668       {%
669         Either remove the automake=true setting or
670         add \string\makeglossaries\space to your document preamble.%
671       }}%
672   }%
673 \else
674   \renewcommand*{\@gls@doautomake}{}%
675 \fi
676 }
677 \glsautomakefalse

```

`@gls@doautomake`

```

678 \newcommand*{\@gls@doautomake}{}
679 \AtEndDocument{\@gls@doautomake}

```

savewrites The savewrites package option is provided to save on the number of write registers.

```
680 \define@boolkey{glossaries.sty}[gls]{savewrites}[true]{%
681   \ifglssavewrites
682     \renewcommand*{\glswritefiles}{\@glswritefiles}%
683   \else
684     \let\glswritefiles\@empty
685   \fi
686 }
```

Set default:

```
687 \glssavewritesfalse
688 \let\glswritefiles\@empty
```

compatible-3.07

```
689 \define@boolkey{glossaries.sty}[gls]{compatible-3.07}[true]{}
690 \boolfalse{glscpatible-3.07}
```

compatible-2.07

```
691 \define@boolkey{glossaries.sty}[gls]{compatible-2.07}[true]{%
  Also set 3.07 compatibility if this option is set.
692   \ifbool{glscpatible-2.07}{%
693     {%
694       \booltrue{glscpatible-3.07}%
695     }%
696   }%
697 }
698 \boolfalse{glscpatible-2.07}
```

symbols Create a “symbols” glossary type

```
699 \@gls@declareoption{symbols}{%
700   \let\@gls@do@symbolsdef\@gls@symbolsdef
701 }
```

Default is not to define the symbols glossary:

```
702 \newcommand*{\@gls@do@symbolsdef}{}%
```

@gls@symbolsdef

```
703 \newcommand*{\@gls@symbolsdef}{%
704   \newglossary[slg]{symbols}{sls}{slo}{\glssymbolsgroupname}%
705   \newcommand*{\printsymbols}[1][\printglossary[type=symbols,##1]]{%
```

Define hook to set the toc title when translator is in use.

```
706   \newcommand*{\gls@tr@set@symbols@toctitle}{%
707     \translatelet{\glossarytoctitle}{Symbols (glossaries)}%
708   }%
709 }
```

numbers Create a “symbols” glossary type

```
710 \@gls@declareoption{numbers}{%  
711   \let\@gls@do@numbersdef\@gls@numbersdef  
712 }
```

Default is not to define the numbers glossary:

```
713 \newcommand*{\@gls@do@numbersdef}{}
```

@gls@numbersdef

```
714 \newcommand*{\@gls@numbersdef}{%  
715   \newglossary[nlg]{numbers}{nls}{nlo}{\glsnumbersgroupname}%  
716   \newcommand*{\printnumbers}[1] [] {\printglossary[type=numbers,##1]}%
```

Define hook to set the toc title when translator is in use.

```
717   \newcommand*{\gls@tr@set@numbers@toctitle}{%  
718     \translatelet{\glossarytoctitle}{Numbers (glossaries)}%  
719   }%  
720 }%
```

index Create an “index” glossary type

```
721 \@gls@declareoption{index}{%  
722   \let\@gls@do@indexdef\@gls@indexdef  
723 }
```

Default is not to define index glossary:

```
724 \newcommand*{\@gls@do@indexdef}{}
```

\@gls@indexdef \indexname isn't set by glossaries.

```
725 \newcommand*{\@gls@indexdef}{%  
726   \newglossary[ilg]{index}{ind}{idx}{\indexname}%  
727   \newcommand*{\printindex}[1] [] {\printglossary[type=index,##1]}%  
728   \newcommand*{\newterm}[2] [] {%  
729     \newglossaryentry{##2}%  
730     {type={index},name={##2},description={\nopostdesc},##1}}  
731 }%
```

Process package options. First process any options that have been passed via the document class.

```
732 \@for\CurrentOption :=\@declaredoptions\do{%  
733   \ifx\CurrentOption\@empty  
734   \else  
735     \@expandtwoargs  
736     \in@ {,\CurrentOption ,}{,\@classoptionslist,\@curroptions,}%  
737     \ifin@  
738     \@use@ption  
739     \expandafter \let\csname ds@\CurrentOption\endcsname\@empty  
740   \fi  
741 \fi  
742 }
```

Now process options passed to the package:

```
743 \ProcessOptionsX
```

Load backward compatibility stuff:

```
744 \RequirePackage{glossaries-compatible-307}
```

`setupglossaries` Provide way to set options after package has been loaded. However, some options must be set before `\ProcessOptionsX`, so they have to be disabled:

```
745 \disable@keys{glossaries.sty}{compatible-2.07,%
```

```
746 xindy,xindygloss,xindynoglsnumbers,makeindex,%
```

```
747 acronym,translate,notranslate,nolong,nosuper,notree,nostyles,nomain}
```

Now define `\setupglossaries`:

```
748 \newcommand*{\setupglossaries}[1]{%
```

```
749 \renewcommand*{\@gls@setacrstyle}{}%
```

```
750 \ifglsacrshortcuts
```

```
751 \def\@gls@setupshortcuts{\glsacrshortcutstrue}%
```

```
752 \else
```

```
753 \def\@gls@setupshortcuts{%
```

```
754 \ifglsacrshortcuts
```

```
755 \DefineAcronymSynonyms
```

```
756 \fi
```

```
757 }%
```

```
758 \fi
```

```
759 \glsacrshortcutsfalse
```

```
760 \let\@gls@do@numbersdef\relax
```

```
761 \let\@gls@do@symbolssdef\relax
```

```
762 \let\@gls@do@indexdef\relax
```

```
763 \let\@gls@do@acronymsdef\relax
```

```
764 \setkeys{glossaries.sty}{#1}%
```

```
765 \@gls@setacrstyle
```

```
766 \@gls@setupshortcuts
```

```
767 \@gls@do@acronymsdef
```

```
768 \@gls@do@numbersdef
```

```
769 \@gls@do@symbolssdef
```

```
770 \@gls@do@indexdef
```

```
771 }
```

If chapters are defined and the user has requested the section counter as a package option, `\@chapter` will be modified so that it adds a section. $\langle n \rangle . 0$ target, otherwise entries placed before the first section of a chapter will have undefined links.

The same problem will also occur if a lower sectional unit is used, but this is less likely to happen. If it does, or if you change `\glscounter` to section later, you will have to specify a different counter for the entries that give rise to a name $\{\langle section-level \rangle . \langle n \rangle . 0\}$ non-existent warning (e.g. `\gls[counter=chapter]{label}`).

```
772 \ifthenelse{\equal{\glscounter}{section}}{%
```

```
773 {%
```

```
774 \ifcsundef{chapter}{}%
```

```
775 {%
```

```

776 \let\@gls@old@chapter\@chapter
777 \def\@chapter[#1]#2{\@gls@old@chapter[#1]{#2}%
778 \ifcsundef{hyperdef}{\hyperdef{section}{\thesection}{}}}%
779 }%
780 }%
781 {}

```

`\ls@onlypremakeg` Some commands only have an effect when used before `\makeglossaries`. So define a list of commands that should be disabled after `\makeglossaries`

```

782 \newcommand*{\@gls@onlypremakeg}{}

```

`\@onlypremakeg` Adds the specified control sequence to the list of commands that must be disabled after `\makeglossaries`.

```

783 \newcommand*{\@onlypremakeg}[1]{%
784 \ifx\@gls@onlypremakeg\@empty
785 \def\@gls@onlypremakeg{#1}%
786 \else
787 \expandafter\toks@ \expandafter{\@gls@onlypremakeg}%
788 \edef\@gls@onlypremakeg{\the\toks@, \noexpand#1}%
789 \fi
790 }

```

`\le@onlypremakeg` Disable all commands listed in `\@gls@onlypremakeg`

```

791 \newcommand*{\@disable@onlypremakeg}{%
792 \@for\@thiscs:=\@gls@onlypremakeg\do{%
793 \expandafter\@disable@premakecs\@thiscs%
794 }}

```

`\sable@premakecs` Disables the given command.

```

795 \newcommand*{\@disable@premakecs}[1]{%
796 \def#1{\PackageError{glossaries}{\string#1\space may only be
797 used before \string\makeglossaries}{You can't use
798 \string#1\space after \string\makeglossaries}}%
799 }

```

1.3 Predefined Text

Set up default textual tags that are used by this package. Some of the names may already be defined (e.g. by) so `\providecommand` is used.

Main glossary title:

`\glossaryname`

```

800 \providecommand*{\glossaryname}{Glossary}

```

The title for the acronym glossary type (which is defined if acronym package option is used) is given by `\acronymname`. If the acronym package option is not used, `\acronymname` won't be used.

`\acronymname`
801 `\providecommand*{\acronymname}{Acronyms}`

`\glsettoctitle` Sets the TOC title for the given glossary.
802 `\newcommand*{\glsettoctitle}[1]{%`
803 `\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}`

The following commands provide text for the headers used by some of the tabular-like glossary styles. Whether or not they get used in the glossary depends on the glossary style.

`\entryname`
804 `\providecommand*{\entryname}{Notation}`

`\descriptionname`
805 `\providecommand*{\descriptionname}{Description}`

`\symbolname`
806 `\providecommand*{\symbolname}{Symbol}`

`\pagelistname`
807 `\providecommand*{\pagelistname}{Page List}`

Labels for `makeindex`'s symbol and number groups:

`\symbolsgroupname`
808 `\providecommand*{\glssymbolsgroupname}{Symbols}`

`\numbersgroupname`
809 `\providecommand*{\glsnnumbersgroupname}{Numbers}`

`\glsppluralsuffix` The default plural is formed by appending `\glsppluralsuffix` to the singular form.
810 `\newcommand*{\glsppluralsuffix}{s}`

`\acrpluralsuffix` Default plural suffix for acronyms
811 `\newcommand*{\glacrpluralsuffix}{\glsppluralsuffix}`

`\supacrpluralsuffix`
812 `\newcommand*{\glupacrpluralsuffix}{\glstextup{\glacrpluralsuffix}}`

`\seename`
813 `\providecommand*{\seename}{see}`

`\andname`
814 `\providecommand*{\andname}{\&}`

Add multi-lingual support. Thanks to everyone who contributed to the translations from both `comp.text.tex` and via email.

eGlossariesLang

```
815 \newcommand*{\RequireGlossariesLang}[1]{%
816   \@ifundefined{ver@glossaries-#1.1df}{\input{glossaries-#1.1df}}{}%
817 }
```

sGlossariesLang

```
818 \newcommand*{\ProvidesGlossariesLang}[1]{%
819   \ProvidesFile{glossaries-#1.1df}%
820 }
```

ssarytocaptions Does nothing if translator hasn't been loaded.

```
821 \newcommand*{\addglossarytocaptions}[1]{}
```

As from v4.12, multilingual support has been split off into independently-maintained language modules.

```
822 \ifglstranslate
```

Load tracklang

```
823 \RequirePackage{tracklang}
```

Load translator if required.

```
824 \@gls@usetranslator
```

If using , \glossaryname should be defined in terms of \translate, but if babel is also loaded, it will redefine \glossaryname whenever the language is set, so override it. (Don't use \addto as doesn't define it.)

```
825 \@ifpackageloaded{translator}
```

```
826 {%
```

If the language options have been specified through the document class, then translator can pick them up. If not, translator will default to English and any language option passed to babel won't be detected, so if \trans@languages is just English and \bbl@loaded isn't simply english, then don't use the translator dictionaries.

```
827 \ifboolexpr
```

```
828 {
```

```
829   test {\ifdefstring{\trans@languages}{English}}
```

```
830   and not
```

```
831   test {\ifdefstring{\bbl@loaded}{english}}
```

```
832 }
```

```
833 {%
```

```
834   \let\glsifusetranslator\@secondoftwo
```

```
835 }%
```

```
836 {%
```

```
837   \usedictionary{glossaries-dictionary}%
```

```
838   \renewcommand*{\addglossarytocaptions}[1]{%
```

```
839     \ifcsundef{captions#1}{}%
```

```
840     {%
```

```
841       \expandafter\let\expandafter\@gls@tmp\csname captions#1\endcsname
```

```
842       \expandafter\toks@\expandafter{\@gls@tmp
```



```

843         \renewcommand*{\glossaryname}{\translate{Glossary}}}%
844     }%
845     \expandafter\edef\csname captions#1\endcsname{\the\toks@}%
846 }%
847 }%
848 }%
849 }%
850 {}%

```

Check for tracked languages

```

851 \AnyTrackedLanguages
852 {%
853     \ForEachTrackedDialect{\this@dialect}{%
854         \IfTrackedLanguageFileExists{\this@dialect}%
855         {glossaries-}% prefix
856         {.ldf}%
857         {%
858             \RequireGlossariesLang{\CurrentTrackedTag}%
859         }%
860         {%
861             \PackageWarningNoLine{glossaries}%
862             {No language module detected for ‘\this@dialect’.\MessageBreak
863             Language modules need to be installed separately.\MessageBreak
864             Please check on CTAN for a bundle called\MessageBreak
865             ‘glossaries-\CurrentTrackedLanguage’ or similar}%
866         }%
867     }%
868 }%
869 {}%

```

if using translator use translator interface.

```

870 \glsifusetranslator
871 {%
872     \renewcommand*{\glssettoctitle}[1]{%
873         \ifcsdef{gls@tr@set@#1@toctitle}%
874         {%
875             \csuse{gls@tr@set@#1@toctitle}%
876         }%
877         {%
878             \def\glossarytoctitle{\csname @glotype@#1@title\endcsname}%
879         }%
880     }%
881     \renewcommand*{\glossaryname}{\translate{Glossary}}}%
882     \renewcommand*{\acronymname}{\translate{Acronyms}}}%
883     \renewcommand*{\entryname}{\translate{Notation (glossaries)}}}%
884     \renewcommand*{\descriptionname}{%
885         \translate{Description (glossaries)}}}%
886     \renewcommand*{\symbolname}{\translate{Symbol (glossaries)}}}%
887     \renewcommand*{\pagelistname}{%
888         \translate{Page List (glossaries)}}}%

```

```

889 \renewcommand*{\glssymbolsgroupname}{%
890 \translate{Symbols (glossaries)}}%
891 \renewcommand*{\glsnumbersgroupname}{%
892 \translate{Numbers (glossaries)}}%
893 }{}%
894 \fi

```

`\nopostdesc` Provide a means to suppress description terminator for a given entry. (Useful for entries with no description.) Has no effect outside the glossaries.

```
895 \DeclareRobustCommand*{\nopostdesc}{}

```

`\@nopostdesc` Suppress next description terminator.

```

896 \newcommand*{\@nopostdesc}{%
897 \let\org@glspostdescription\glspostdescription
898 \def\glspostdescription{%
899 \let\glspostdescription\org@glspostdescription}%
900 }

```

`\@no@post@desc` Used for comparison purposes.

```
901 \newcommand*{\@no@post@desc}{\nopostdesc}

```

`\glspar` Provide means of having a paragraph break in glossary entries

```
902 \newcommand{\glspar}{\par}

```

`\setStyleFile` Sets the style file. The relevant extension is appended.

```

903 \newcommand{\setStyleFile}[1]{%
904 \renewcommand*{\gls@istfilebase}{#1}%
  Just in case \istfilename has been modified.
905 \ifglsxindy
906 \def\istfilename{\gls@istfilebase.xdy}
907 \else
908 \def\istfilename{\gls@istfilebase.ist}
909 \fi
910 }

```

This command only has an effect prior to using `\makeglossaries`.

```
911 \@onlypremakeg\setStyleFile

```

The name of the makeindex or xindy style file is given by `\istfilename`. This file is created by `\writeist` (which is used by `\makeglossaries`) so redefining this command will only have an effect if it is done *before* `\makeglossaries`. As from v1.17, use `\setStyleFile` instead of directly redefining `\istfilename`.

`\istfilename`

```

912 \ifglsxindy
913 \def\istfilename{\gls@istfilebase.xdy}
914 \else
915 \def\istfilename{\gls@istfilebase.ist}
916 \fi

```

gls@istfilebase

```
917 \newcommand*{\gls@istfilebase}{\jobname}
```

The `makeglossaries` Perl script picks up this name from the auxiliary file. If the name ends with `.xdy` it calls `xindy` otherwise it calls `makeindex`. Since its not required by \LaTeX , `\@istfilename` ignores its argument.

`\@istfilename`

```
918 \newcommand*{\@istfilename}[1]{}
```

This command is the value of the `page_compositor` `makeindex` key. Again, any redefinition of this command must take place *before* `\writeist` otherwise it will have no effect. As from 1.17, use `\glsSetCompositor` instead of directly redefining `\glscompositor`.

`\glscompositor`

```
919 \newcommand*{\glscompositor}{.}
```

`lsSetCompositor` Sets the compositor.

```
920 \newcommand*{\glsSetCompositor}[1]{%
921   \renewcommand*{\glscompositor}{#1}}
```

Only use before `\makeglossaries`

```
922 \@onlypremakeg\glsSetCompositor
```

(The page compositor is usually defined as a dash when using `makeindex`, but most of the standard counters used by \LaTeX use a full stop as the compositor, which is why I have used it as the default.) If `xindy` is used `\glscompositor` only affects the `arabic-page-numbers` location class.

`Alphacompositor` This is only used by `xindy`. It specifies the compositor to use when location numbers are in the form `<letter><compositor><number>`. For example, if `\@glsAlphacompositor` is set to `."` then it allows locations such as `A.1` whereas if `\@glsAlphacompositor` is set to `"-`" then it allows locations such as `A-1`.

```
923 \newcommand*{\@glsAlphacompositor}{\glscompositor}
```

`AlphaCompositor` Sets the alpha compositor.

```
924 \ifglsxindy
925   \newcommand*\glsSetAlphaCompositor[1]{%
926     \renewcommand*\@glsAlphacompositor{#1}}
927 \else
928   \newcommand*\glsSetAlphaCompositor[1]{%
929     \glsnoindywarning\glsSetAlphaCompositor}
930 \fi
```

Can only be used before `\makeglossaries`

```
931 \@onlypremakeg\glsSetAlphaCompositor
```

`\gls@suffixF` Suffix to use for a two page list. This overrides the separator and the closing page number if set to something other than an empty macro.

```
932 \newcommand*{\gls@suffixF}{}%
```

`\glsSetSuffixF` Sets the suffix to use for a two page list.

```

933 \newcommand*{\glsSetSuffixF}[1]{%
934   \renewcommand*{\gls@suffixF}{#1}}

```

Only has an effect when used before `\makeglossaries`

```

935 \@onlypremakeg\glsSetSuffixF

```

`\gls@suffixFF` Suffix to use for a three page list. This overrides the separator and the closing page number if set to something other than an empty macro.

```

936 \newcommand*{\gls@suffixFF}{}

```

`\glsSetSuffixFF` Sets the suffix to use for a three page list.

```

937 \newcommand*{\glsSetSuffixFF}[1]{%
938   \renewcommand*{\gls@suffixFF}{#1}%
939 }

```

`\glsnumberformat` The command `\glsnumberformat` indicates the default format for the page numbers in the glossary. (Note that this is not the same as `\glossaryentrynumbers`, but applies to individual numbers or groups of numbers within an entry's associated number list.) If hyperlinks are defined, it will use `\glshypernumber`, otherwise it will simply display its argument "as is".

```

940 \ifcsundef{hyperlink}%
941 {%
942   \newcommand*{\glsnumberformat}[1]{#1}%
943 }%
944 {%
945   \newcommand*{\glsnumberformat}[1]{\glshypernumber{#1}}%
946 }

```

Individual numbers in an entry's associated number list are delimited using `\delimN` (which corresponds to the `delim_n` `makeindex` keyword). The default value is a comma followed by a space.

`\delimN`

```

947 \newcommand{\delimN}{, }

```

A range of numbers within an entry's associated number list is delimited using `\delimR` (which corresponds to the `delim_r` `makeindex` keyword). The default is an en-dash.

`\delimR`

```

948 \newcommand{\delimR}{--}

```

The glossary preamble is given by `\glossarypreamble`. This will appear after the glossary sectioning command, and before the `\theglossary` environment. It is designed to allow the user to add information pertaining to the glossary (e.g. "page numbers in italic indicate the primary definition") therefore `\glossarypreamble` shouldn't be affected by the glossary style. (So if you define your own glossary style, don't have it change `\glossarypreamble`.)

The preamble is empty by default. If you have multiple glossaries, and you want a different preamble for each glossary, you will need to use `\printglossary` for each glossary type, instead of `\printglossaries`, and redefine `\glossarypreamble` before each `\printglossary`.

`\glossarypreamble`

```
949 \newcommand*{\glossarypreamble}{%
950   \csuse{@glossarypreamble@\currentglossary}%
951 }
```

`\glossarypreamble`

```
\setglossarypreamble[<type>]{<text>}
```

Code provided by Michael Pock.

```
952 \newcommand{\setglossarypreamble}[2][\glsdefaulttype]{%
953   \ifglossaryexists{#1}{%
954     \csgdef{@glossarypreamble@#1}{#2}%
955   }{%
956     \GlossariesWarning{%
957       Glossary ‘#1’ is not defined%
958     }%
959   }%
960 }
```

The glossary postamble is given by `\glossarypostamble`. This is provided to allow the user to add something after the end of the `\glossary` environment (again, this shouldn't be affected by the glossary style). It is, of course, possible to simply add the text after `\printglossary`, but if you only want the postamble to appear after the first glossary, but not after subsequent glossaries, you can do something like:

```
\renewcommand{\glossarypostamble}{For a complete list of terms
see \cite{blah}\gdef\glossarypreamble{}}
```

`\glossarypostamble`

```
961 \newcommand*{\glossarypostamble}{}
```

`\glossarysection`

The sectioning command that starts a glossary is given by `\glossarysection`. (This does not form part of the glossary style, and so should not be changed by a glossary style.) If `\phantomsection` is defined, it uses `\p@glossarysection`, otherwise it uses `\@glossarysection`.

```
962 \newcommand*{\glossarysection}[2][\@gls@title]{%
963   \def\@gls@title{#2}%
964   \ifcsundef{phantomsection}%
965   {%
966     \@glossarysection{#1}{#2}%
967   }%
968   {%
969     \p@glossarysection{#1}{#2}%
970   }%
```

```

971 \glsglossarymark{\glossarytoctitle}%
972 }

```

`glsglossarymark` Sets the header mark for the glossary. Takes the glossary short (TOC) title as the argument.

```

973 \ifcsundef{glossarymark}%
974 {%
975   \newcommand{\glsglossarymark}[1]{\glossarymark{#1}}
976 }%
977 {%
978   \@ifclassloaded{memoir}
979   {%
980     \newcommand{\glsglossarymark}[1]{%
981       \ifglsucmark
982         \markboth{\memUHead{#1}}{\memUHead{#1}}%
983       \else
984         \markboth{#1}{#1}%
985       \fi
986     }
987   }%
988   {%
989     \newcommand{\glsglossarymark}[1]{%
990       \ifglsucmark
991         \@mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
992       \else
993         \@mkboth{#1}{#1}%
994       \fi
995     }
996   }
997 }

```

`\glossarymark` Provided for backward compatibility:

```

998 \providecommand{\glossarymark}[1]{%
999   \ifglsucmark
1000     \@mkboth{\mfirstucMakeUppercase{#1}}{\mfirstucMakeUppercase{#1}}%
1001   \else
1002     \@mkboth{#1}{#1}%
1003   \fi
1004 }

```

The required sectional unit is given by `\@glossarysec` which was defined by the section package option. The starred form of the command is chosen. If you don't want any sectional command, you will need to redefine `\glossarysection`. The sectional unit can be changed, if different sectional units are required.

`glossarysection`

```

1005 \newcommand*{\setglossarysection}[1]{%
1006 \setkeys{glossaries.sty}{section=#1}}

```

The command `\@glossarysection` indicates how to start the glossary section if `\phantomsection` is not defined.

glossarysection

```

1007 \newcommand*{\@glossarysection}[2]{%
1008   \ifdefempty\@glossarysecstar
1009   {%
1010     \csname\@glossarysec\endcsname[#1]{#2}%
1011   }%
1012   {%
1013     \csname\@glossarysec\endcsname*{#2}%
1014     \@gls@toc{#1}{\@glossarysec}%
1015   }%

  Do automatic labelling if required
1016   \@glossaryseclabel
1017 }

```

As \@glossarysection, but put in \phantomsection, and swap where \@gls@toc goes. If using chapters do a \clearpage. This ensures that the hyper link from the table of contents leads to the line above the heading, rather than the line below it.

glossarysection

```

1018 \newcommand*{\@p@glossarysection}[2]{%
1019   \glsclearpage
1020   \phantomsection
1021   \ifdefempty\@glossarysecstar
1022   {%
1023     \csname\@glossarysec\endcsname{#2}%
1024   }%
1025   {%
1026     \@gls@toc{#1}{\@glossarysec}%
1027     \csname\@glossarysec\endcsname*{#2}%
1028   }%

  Do automatic labelling if required
1029   \@glossaryseclabel
1030 }

```

\gls@doclearpage The \gls@doclearpage command is used to issue a \clearpage (or \cleardoublepage) depending on whether the glossary sectional unit is a chapter. If the sectional unit is something else, do nothing.

```

1031 \newcommand*{\gls@doclearpage}{%
1032   \ifthenelse{\equal{\@glossarysec}{chapter}}{%
1033     {%
1034       \ifcsundef{cleardoublepage}%
1035       {%
1036         \clearpage
1037       }%
1038     }%
1039     \ifcsdef{if@openright}%
1040     {%
1041       \if@openright

```

```

1042         \cleardoublepage
1043     \else
1044         \clearpage
1045     \fi
1046 }%
1047 {%
1048     \cleardoublepage
1049 }%
1050 }%
1051 }%
1052 {}%
1053 }

```

`\glsclearpage` This just calls `\gls@doclearpage`, but it makes it easier to have a user command so that the user can override it.

```

1054 \newcommand*\glsclearpage{\gls@doclearpage}

```

The glossary is added to the table of contents if `glstoc` flag set. If it is set, `\@gls@toc` will add a line to the `.toc` file, otherwise it will do nothing. (The first argument to `\@gls@toc` is the title for the table of contents, the second argument is the sectioning type.)

`\@gls@toc`

```

1055 \newcommand*\@gls@toc[2]{%
1056     \ifglstoc
1057         \ifglsnumberline
1058             \addcontentsline{toc}{#2}{\protect\numberline{#1}}%
1059         \else
1060             \addcontentsline{toc}{#2}{#1}%
1061         \fi
1062     \fi
1063 }

```

1.4 Xindy

This section defines commands that only have an effect if `xindy` is used to sort the glossaries.

`\glsnnoxindywarning` Issues a warning if `xindy` hasn't been specified. These warnings can be suppressed by re-defining `\glsnnoxindywarning` to ignore its argument

```

1064 \newcommand*\glsnnoxindywarning[1]{%
1065     \GlossariesWarning{Not in xindy mode --- ignoring \string#1}%
1066 }

```

`\glsnomakeindexwarning` Reverse for commands that may only be used with `makeindex`.

```

1067 \newcommand*\glsnomakeindexwarning[1]{%
1068     \GlossariesWarning{Not in makeindex mode --- ignoring \string#1}%
1069 }

```


`\@xdyattributes` Define list of attributes (`\string` is used in case the double quote character has been made active)

```
1070 \ifglxsindy
1071   \edef\@xdyattributes{\string"default\string"}%
1072 \fi
```

`\@dyattributelist` Comma-separated list of attributes.

```
1073 \ifglxsindy
1074   \edef\@dyattributelist{}%
1075 \fi
```

`\@xdylocref` Define list of markup location references.

```
1076 \ifglxsindy
1077   \def\@xdylocref{}
1078 \fi
```

`\@gls@ifinlist`

```
1079 \newcommand*{\@gls@ifinlist}[4]{%
1080   \def\@do@ifinlist##1,#1,##2\end@ifinlist{%
1081     \def\@gls@listsuffix{##2}%
1082     \ifx\@gls@listsuffix\@empty
1083       #4%
1084     \else
1085       #3%
1086     \fi
1087   }%
1088   \@do@ifinlist,##2,#1,\end@ifinlist
1089 }
```

`\@sAddXdyCounters` Need to know all the counters that will be used in location numbers for Xindy. Argument may be a single counter name or a comma-separated list of counter names.

```
1090 \ifglxsindy
1091   \newcommand*{\@xdycounters}{\@glscounter}
1092   \newcommand*\GlsAddXdyCounters[1]{%
1093     \@for\@gls@ctr:=#1\do{%
1094       Check if already in list before adding.
1095       \edef\@do@addcounter{%
1096         \noexpand\@gls@ifinlist{\@gls@ctr}{\@xdycounters}{}%
1097         \noexpand\edef\noexpand\@xdycounters{\@xdycounters,%
1098           \noexpand\@gls@ctr}%
1099       }%
1100     }%
1101     \@do@addcounter
1102   }
1103 }
```

Only has an effect before `\writeist`:

```

1104 \@onlypremakeg\GlsAddXdyCounters
1105 \else
1106   \newcommand*\GlsAddXdyCounters[1]{%
1107     \glsnxindywarning\GlsAddXdyAttribute
1108   }
1109 \fi

```

`saddxdycounters` Counters must all be identified before adding attributes.

```

1110 \newcommand*\@disabled@glssaddxdycounters{%
1111   \PackageError{glossaries}{\string\GlsAddXdyCounters\space
1112     can't be used after \string\GlsAddXdyAttribute}{Move all
1113     occurrences of \string\GlsAddXdyCounters\space before the first
1114     instance of \string\GlsAddXdyAttribute}%
1115 }

```

`AddXdyAttribute` Adds an attribute.

```

1116 \ifglxsindy

```

First define internal command that adds an attribute for a given counter (2nd argument is the counter):

```

1117 \newcommand*\@glssaddxdyattribute[2]{%

```

Add to xindy attribute list

```

1118   \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string" ^^J
1119     \string"#2#1\string"}%

```

Add to xindy markup location.

```

1120   \expandafter\toks@\expandafter{\@xdylocref}%
1121   \edef\@xdylocref{\the\toks@ ^^J%
1122     (markup-locref
1123     :open \string"\glstildechar n%
1124       \expandafter\string\csname glsX#2X#1\endcsname
1125       \string" ^^J
1126     :close \string"\string" ^^J
1127     :attr \string"#2#1\string")}%

```

Define associated attribute command `\glsX<counter>X<attribute>{\<Hprefix>}{\<n>}`

```

1128   \expandafter\gdef\csname glsX#2X#1\endcsname##1##2{%
1129     \setentrycounter[##1]{#2}\csname #1\endcsname{##2}%
1130   }%
1131 }

```

High-level command:

```

1132 \newcommand*\GlsAddXdyAttribute[1]{%

```

Add to comma-separated attribute list

```

1133   \ifx\@xdyattributelist\@empty
1134     \edef\@xdyattributelist{#1}%
1135   \else
1136     \edef\@xdyattributelist{\@xdyattributelist,#1}%
1137   \fi

```

Iterate through all specified counters and add counter-dependent attributes:

```

1138 \for\@this@counter:=\@xdycounters\do{%
1139 \protected@edef\gls@do@addxdyattribute{%
1140 \noexpand\@glsaddxdyattribute{#1}{\@this@counter}%
1141 }
1142 \gls@do@addxdyattribute
1143 }%

```

All occurrences of `\GlsAddXdyCounters` must be used before this command

```

1144 \let\GlsAddXdyCounters\@disabled@glsaddxdycounters
1145 }

```

Only has an effect before `\writeist`:

```

1146 \@onlypremakeg\GlsAddXdyAttribute
1147 \else
1148 \newcommand*\GlsAddXdyAttribute[1]{%
1149 \glsnoxywarning\GlsAddXdyAttribute}
1150 \fi

```

`\definedattributes` Add known attributes for all defined counters

```

1151 \ifglxindy
1152 \newcommand*\@gls@addpredefinedattributes{%
1153 \GlsAddXdyAttribute{glsnumberformat}
1154 \GlsAddXdyAttribute{textrm}
1155 \GlsAddXdyAttribute{textsf}
1156 \GlsAddXdyAttribute{texttt}
1157 \GlsAddXdyAttribute{textbf}
1158 \GlsAddXdyAttribute{textmd}
1159 \GlsAddXdyAttribute{textit}
1160 \GlsAddXdyAttribute{textup}
1161 \GlsAddXdyAttribute{textsl}
1162 \GlsAddXdyAttribute{textsc}
1163 \GlsAddXdyAttribute{emph}
1164 \GlsAddXdyAttribute{glshypernumber}
1165 \GlsAddXdyAttribute{hyperrm}
1166 \GlsAddXdyAttribute{hypersf}
1167 \GlsAddXdyAttribute{hypertt}
1168 \GlsAddXdyAttribute{hyperbf}
1169 \GlsAddXdyAttribute{hypermd}
1170 \GlsAddXdyAttribute{hyperit}
1171 \GlsAddXdyAttribute{hyperup}
1172 \GlsAddXdyAttribute{hypersl}
1173 \GlsAddXdyAttribute{hypersc}
1174 \GlsAddXdyAttribute{hyperemph}

1175 \GlsAddXdyAttribute{glsignore}
1176 }
1177 \else
1178 \let\@gls@addpredefinedattributes\relax
1179 \fi

```

dyuseralphabets List of additional alphabets

```
1180 \def\@xdyuseralphabets{}
```

sAddXdyAlphabet \GlsAddXdyAlphabet{<name>}{<definition>} adds a new alphabet called <name>. The definition must use xindy syntax.

```
1181 \ifglxsindy
1182   \newcommand*{\GlsAddXdyAlphabet}[2]{%
1183     \edef\@xdyuseralphabets{%
1184       \@xdyuseralphabets ^^J
1185       (define-alphabet "#1" (#2))}%
1186   \else
1187     \newcommand*{\GlsAddXdyAlphabet}[2]{%
1188       \glsnnoxindywarning\GlsAddXdyAlphabet}
1189 \fi
```

This code is only required for xindy:

```
1190 \ifglxsindy
```

dy@locationlist List of predefined location names.

```
1191   \newcommand*{\@gls@xdy@locationlist}{%
1192     roman-page-numbers,%
1193     Roman-page-numbers,%
1194     arabic-page-numbers,%
1195     alpha-page-numbers,%
1196     Alpha-page-numbers,%
1197     Appendix-page-numbers,%
1198     arabic-section-numbers%
1199   }
```

Each location class <name> has the format stored in \@gls@xdy@Lclass@<name>. Set up predefined formats.

an-page-numbers Lower case Roman numerals (i, ii, ...). In the event that \roman has been redefined to produce a fancy form of roman numerals, attempt to work out how it will be written to the output file.

```
1200   \protected@edef\@gls@roman{\@roman{0}\string"
1201     \string"roman-numbers-lowercase\string" :sep \string"}}%
1202   \@onelevel@sanitize\@gls@roman
1203   \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
1204     :sep \string"}%
1205   \@onelevel@sanitize\@tmp
1206   \ifx\@tmp\@gls@roman
1207     \expandafter
1208       \edef\csname @gls@xdy@Lclass@roman-page-numbers\endcsname{%
1209         \string"roman-numbers-lowercase\string"%
1210       }%
1211   \else
1212     \expandafter
```

```

1213      \edef\csname @gls@xdy@Lclass@roman-page-numbers\endcsname{
1214      :sep \string"\@gls@roman\string"%
1215      }%
1216 \fi

```

an-page-numbers Upper case Roman numerals (I, II, ...).

```

1217 \expandafter\def\csname @gls@xdy@Lclass@Roman-page-numbers\endcsname{%
1218 \string"roman-numbers-uppercase\string"%
1219 }%

```

ic-page-numbers Arabic numbers (1, 2, ...).

```

1220 \expandafter\def\csname @gls@xdy@Lclass@arabic-page-numbers\endcsname{%
1221 \string"arabic-numbers\string"%
1222 }%

```

ha-page-numbers Lower case alphabetical (a, b, ...).

```

1223 \expandafter\def\csname @gls@xdy@Lclass@alpha-page-numbers\endcsname{%
1224 \string"alpha\string"%
1225 }%

```

ha-page-numbers Upper case alphabetical (A, B, ...).

```

1226 \expandafter\def\csname @gls@xdy@Lclass@Alpha-page-numbers\endcsname{%
1227 \string"ALPHA\string"%
1228 }%

```

ix-page-numbers Appendix style locations (e.g. A-1, A-2, ..., B-1, B-2, ...). The separator is given by \@glsAlphacompositor.

```

1229 \expandafter\def\csname @gls@xdy@Lclass@Appendix-page-numbers\endcsname{%
1230 \string"ALPHA\string"
1231 :sep \string"\@glsAlphacompositor\string"
1232 \string"arabic-numbers\string"%
1233 }

```

section-numbers Section number style locations (e.g. 1.1, 1.2, ...). The compositor is given by \glscompositor.

```

1234 \expandafter\def\csname @gls@xdy@Lclass@arabic-section-numbers\endcsname{%
1235 \string"arabic-numbers\string"
1236 :sep \string"\glscompositor\string"
1237 \string"arabic-numbers\string"%
1238 }%

```

erlocationdefs List of additional location definitions (separated by ^^J)

```

1239 \def\@xdyuserlocationdefs{}

```

erlocationnames List of additional user location names

```

1240 \def\@xdyuserlocationnames{}

```

End of xindy-only block:

```

1241 \fi

```

`\GlsAddXdyLocation` [*<prefix-loc>*] {*<name>*} {*<definition>*} Define a new location called *<name>*. The definition must use xindy syntax. (Note that this doesn't check to see if the location is already defined. That is left to xindy to complain about.)

```

1242 \ifglxsindy
1243   \newcommand*{\GlsAddXdyLocation}[3][\]{%
1244     \def\@gls@tmp{#1}%
1245     \ifx\@gls@tmp\@empty
1246       \edef\@xdyuserlocationdefs{%
1247         \@xdyuserlocationdefs ^^J%
1248         (define-location-class \string"#2\string"^^J\space\space
1249         \space(:sep \string"{}\glssopenbrace\string" #3
1250         :sep \string"\glsclosebrace\string"))
1251       }%
1252     \else
1253       \edef\@xdyuserlocationdefs{%
1254         \@xdyuserlocationdefs ^^J%
1255         (define-location-class \string"#2\string"^^J\space\space
1256         \space(:sep "\glssopenbrace"
1257         #1
1258         :sep "\glsclosebrace\glssopenbrace" #3
1259         :sep "\glsclosebrace"))
1260       }%
1261     \fi
1262     \edef\@xdyuserlocationnames{%
1263       \@xdyuserlocationnames^^J\space\space\space
1264       \string"#1\string"}%
1265   }

```

Only has an effect before `\writeist`:

```

1266 \@onlypremakeg\GlsAddXdyLocation
1267 \else
1268   \newcommand*{\GlsAddXdyLocation}[2]{%
1269     \glsnnoxindywarning\GlsAddXdyLocation}
1270 \fi

```

`\locationclassorder` Define location class order

```

1271 \ifglxsindy
1272   \edef\@xdylocationclassorder{^^J\space\space\space
1273     \string"roman-page-numbers\string"^^J\space\space\space
1274     \string"arabic-page-numbers\string"^^J\space\space\space
1275     \string"arabic-section-numbers\string"^^J\space\space\space
1276     \string"alpha-page-numbers\string"^^J\space\space\space
1277     \string"Roman-page-numbers\string"^^J\space\space\space
1278     \string"Alpha-page-numbers\string"^^J\space\space\space
1279     \string"Appendix-page-numbers\string"
1280     \@xdyuserlocationnames^^J\space\space\space
1281     \string"see\string"
1282   }
1283 \fi

```

Change the location order.

ationClassOrder

```
1284 \ifglxindy
1285   \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1286     \def\@xdylocationclassorder{#1}}
1287 \else
1288   \newcommand*\GlsSetXdyLocationClassOrder[1]{%
1289     \glsnoxywarning\GlsSetXdyLocationClassOrder}
1290 \fi
```

\@xdysortrules Define sort rules

```
1291 \ifglxindy
1292   \def\@xdysortrules{}
1293 \fi
```

\GlsAddSortRule Add a sort rule

```
1294 \ifglxindy
1295   \newcommand*\GlsAddSortRule[2]{%
1296     \expandafter\toks@\expandafter{\@xdysortrules}%
1297     \protected@edef\@xdysortrules{\the\toks@ ^^J
1298       (sort-rule \string"#1\string" \string"#2\string")}%
1299   }
1300 \else
1301   \newcommand*\GlsAddSortRule[2]{%
1302     \glsnoxywarning\GlsAddSortRule}
1303 \fi
```

yrequiredstyles Define list of required styles (this should be a comma-separated list of xindy styles)

```
1304 \ifglxindy
1305   \def\@xdyrequiredstyles{tex}
1306 \fi
```

\GlsAddXdyStyle Add a xindy style to the list of required styles

```
1307 \ifglxindy
1308   \newcommand*\GlsAddXdyStyle[1]{%
1309     \edef\@xdyrequiredstyles{\@xdyrequiredstyles,#1}}%
1310 \else
1311   \newcommand*\GlsAddXdyStyle[1]{%
1312     \glsnoxywarning\GlsAddXdyStyle}
1313 \fi
```

GlsSetXdyStyles Reset the list of required styles

```
1314 \ifglxindy
1315   \newcommand*\GlsSetXdyStyles[1]{%
1316     \edef\@xdyrequiredstyles{#1}}
1317 \else
1318   \newcommand*\GlsSetXdyStyles[1]{%
1319     \glsnoxywarning\GlsSetXdyStyles}
1320 \fi
```

`\findrootlanguage` This used to determine the root language, using a bit of trickery since babel doesn't supply the information, but now that babel is once again actively maintained, we can't do this any more, so `\findrootlanguage` is no longer available. Now provide a command that does nothing (in case it's been patched), but this may be removed completely in the future.

```
1321 \newcommand*\findrootlanguage{}
```

`\@xdylanguage` The xindy language setting is required by `makeglossaries`, so provide a command for `makeglossaries` to pick up the information from the auxiliary file. This command is not needed by the `glossaries` package, so define it to ignore its arguments.

```
1322 \def\@xdylanguage#1#2{}
```

`\GlsSetXdyLanguage` Define a command that allows the user to set the language for a given glossary type. The first argument indicates the glossary type. If omitted the main glossary is assumed.

```
1323 \ifglxindy
1324   \newcommand*\GlsSetXdyLanguage[2][\glsdefaulttype]{%
1325     \ifglossaryexists{#1}{%
1326       \expandafter\def\csname @xdy@#1@language\endcsname{#2}%
1327     }{%
1328       \PackageError{glossaries}{Can't set language type for
1329         glossary type '#1' --- no such glossary}{%
1330         You have specified a glossary type that doesn't exist}}
1331 \else
1332   \newcommand*\GlsSetXdyLanguage[2][]{%
1333     \glsnoxywarning\GlsSetXdyLanguage}
1334 \fi
```

`\@gls@codepage` The xindy codepage setting is required by `makeglossaries`, so provide a command for `makeglossaries` to pick up the information from the auxiliary file. This command is not needed by the `glossaries` package, so define it to ignore its arguments.

```
1335 \def\@gls@codepage#1#2{}
```

`\GlsSetXdyCodePage` Define command to set the code page.

```
1336 \ifglxindy
1337   \newcommand*\GlsSetXdyCodePage[1]{%
1338     \renewcommand*\@gls@codepage{#1}%
1339   }
```

Suggested by egreg:

```
1340   \AtBeginDocument{%
1341     \ifx\@gls@codepage\@empty
1342       \@ifpackageloaded{fontspec}{\def\@gls@codepage{utf8}}{}%
1343     \fi
1344   }
1345 \else
1346   \newcommand*\GlsSetXdyCodePage[1]{%
1347     \glsnoxywarning\GlsSetXdyCodePage}
1348 \fi
```


`\xdylettergroups` Store letter group definitions.

```

1349 \ifglxsindy
1350   \ifglxs@xindy@glnumbers
1351     \def\xdylettergroups{(define-letter-group
1352       \string"glnumbers\string"^^J\space\space\space
1353       :prefixes (\string"0\string" \string"1\string"
1354       \string"2\string" \string"3\string" \string"4\string"
1355       \string"5\string" \string"6\string" \string"7\string"
1356       \string"8\string" \string"9\string")^^J\space\space\space
1357       :before \string"@glfirstletter\string")}
1358   \else
1359     \def\xdylettergroups{}
1360   \fi
1361 \fi

```

`\sAddLetterGroup` Add a new letter group. The first argument is the name of the letter group. The second argument is the xindy code specifying prefixes and ordering.

```

1362   \newcommand*\GlsAddLetterGroup[2]{%
1363     \expandafter\toks@\expandafter{\@xdylettergroups}%
1364     \protected@edef\xdylettergroups{\the\toks@^^J%
1365     (define-letter-group \string"#1\string"^^J\space\space\space#2)}%
1366   }%

```

1.5 Loops and conditionals

`\forall glossaries` To iterate through all glossaries (or comma-separated list of glossary names given in optional argument) use:

```
\forall glossaries[<glossary list>]{<cmd>}{<code>}
```

where *<cmd>* is a control sequence which will be set to the name of the glossary in the current iteration.

```

1367 \newcommand*\forallglossaries}[3][\@glo@types]{%
1368   \@for#2:=#1\do{\ifx#2\@empty\else#3\fi}%
1369 }

```

`\forall acronyms`

```

1370 \newcommand*\forallacronyms}[2]{%
1371   \@for#1:=\@glsacronymlists\do{\ifx#1\@empty\else#2\fi}%
1372 }

```

`\forall sentries` To iterate through all entries in a given glossary use:

```
\forall sentries[<type>]{<cmd>}{<code>}
```

where *<type>* is the glossary label and *<cmd>* is a control sequence which will be set to the entry label in the current iteration.

```

1373 \newcommand*{\forglentries}[3][\glsdefaulttype]{%
1374   \edef\@glo@list{\csname glolist@#1\endcsname}%
1375   \@for#2:=\@glo@list\do
1376   {%
1377     \ifdefempty{#2}{\#{#3}%
1378     }%
1379 }

```

`\forallglentries` To iterate through all glossary entries over all glossaries listed in the optional argument (the default is all glossaries) use:

```
\forallglentries[<glossary list>]{<cmd>}{<code>}
```

Within `\forallglentries`, the current glossary type is given by `\@thisglo@`.

```

1380 \newcommand*{\forallglentries}[3][\@glo@types]{%
1381   \expandafter\forallglossaries\expandafter[#1]{\@thisglo@}%
1382   {%
1383     \forglentries[\@thisglo@]{#2}{#3}%
1384   }%
1385 }

```

`\ifglossaryexists` To check to see if a glossary exists use:

```
\ifglossaryexists{<type>}{<true-text>}{<false-text>}
```

where *<type>* is the glossary's label.

```

1386 \newcommand{\ifglossaryexists}[3]{%
1387   \ifcsundef{@glo@type@#1@out}{#3}{#2}%
1388 }

```

Since the label is used to form the name of control sequences, by default UTF8 etc characters can't be used in the label. A possible workaround is to use `\scantokens`, but commands such as `\glsentrytext` will no longer be usable in sectioning, caption etc commands. If the user really wants to be able to construct a label with UTF8 characters, allow them the means to do so (but on their own head be it, if they then use entries in `\section` etc). This can be done via:

```
\renewcommand*{\glsdetoklabel}[1]{\scantokens{#1\noexpand}}
```

(Note, don't use `\detokenize` or it will cause commands like `\glsaddall` to fail.) Since re-defining `\glsdetoklabel` can cause things to go badly wrong, I'm not going to mention it in the main user guide. Only advanced users who know what they're doing ought to attempt it.

`\glsdetoklabel`

```
1389 \newcommand*{\glsdetoklabel}[1]{#1}
```

`\ifglentryexists` To check to see if a glossary entry has been defined use:

```
\ifglentryexists{<label>}{<true text>}{<false text>}
```

where $\langle label \rangle$ is the entry's label.

```
1390 \newcommand{\ifglentryexists}[3]{%
1391   \ifcsundef{glo@\glsdetoklabel{#1}@name}{#3}{#2}%
1392 }
```

\backslash ifglused To determine if given glossary entry has been used in the document text yet use:

```
\ifglused{<label>}{<true text>}{<false text>}
```

where $\langle label \rangle$ is the entry's label. If true it will do $\langle true text \rangle$ otherwise it will do $\langle false text \rangle$.

```
1393 \newcommand*{\ifglused}[3]{%
1394   \ifbool{glo@\glsdetoklabel{#1}@flag}{#2}{#3}%
1395 }
```

The following two commands will cause an error if the given condition fails:

\backslash glsdoifexists

```
\glsdoifexists{<label>}{<code>}
```

Generate an error if entry specified by $\langle label \rangle$ doesn't exists, otherwise do $\langle code \rangle$.

```
1396 \newcommand{\glsdoifexists}[2]{%
1397   \ifglentryexists{#1}{#2}{%
1398     \PackageError{glossaries}{Glossary entry '\glsdetoklabel{#1}'
1399     has not been defined}{You need to define a glossary entry before you
1400     can use it.}}%
1401 }
```

\backslash glsdoifnoexists

```
\glsdoifnoexists{<label>}{<code>}
```

The opposite: only do second argument if the entry doesn't exists. Generate an error message if it exists.

```
1402 \newcommand{\glsdoifnoexists}[2]{%
1403   \ifglentryexists{#1}{#2}{%
1404     \PackageError{glossaries}{Glossary entry '\glsdetoklabel{#1}' has already
1405     been defined}{}}{#2}%
1406 }
```

\backslash glsdoifexistsorwarn

```
\glsdoifexistsorwarn{<label>}{<code>}
```

Generate a warning if entry specified by $\langle label \rangle$ doesn't exists, otherwise do $\langle code \rangle$.

```
1407 \newcommand{\glsdoifexistsorwarn}[2]{%
1408   \ifglentryexists{#1}{#2}{%
1409     \GlossariesWarning{Glossary entry '\glsdetoklabel{#1}'
1410     has not been defined}%
1411   }%
1412 }
```

glsdoifexistsordo `\glsdoifexistsordo{<label>}{<code>}{<undef code>}`

Generate an error and do *<undef code>* if entry specified by *<label>* doesn't exist, otherwise do *<code>*.

```

1413 \newcommand{\glsdoifexistsordo}[3]{%
1414   \ifglentryexists{#1}{#2}{%
1415     \PackageError{glossaries}{Glossary entry ‘\glsdetoklabel{#1}’
1416       has not been defined}{You need to define a glossary entry before you
1417       can use it.}%
1418     #3%
1419   }%
1420 }
```

sarynoexistsordo `\doifglossarynoexistsordo{<label>}{<code>}{<else code>}`

If glossary given by *<label>* doesn't exist do *<code>* otherwise generate an error and do *<else code>*.

```

1421 \newcommand{\doifglossarynoexistsordo}[3]{%
1422   \ifglossaryexists{#1}%
1423   {%
1424     \PackageError{glossaries}{Glossary type ‘#1’ already exists}{}%
1425     #3%
1426   }%
1427   {#2}%
1428 }
```

glshaschildren `\ifglshaschildren{<label>}{<true part>}{<false part>}`

```

1429 \newcommand{\ifglshaschildren}[3]{%
1430   \glsdoifexists{#1}%
1431   {%
1432     \def\do@glshaschildren{#3}%
1433     \edef\@gls@thislabel{\glsdetoklabel{#1}}%
1434     \expandafter\forglentries\expandafter
1435     [\csname glo@\@gls@thislabel @type\endcsname]
1436     {\glo@label}%
1437     {%
1438       \letcs\glo@parent{glo@\glo@label @parent}%
1439       \ifdefequal\@gls@thislabel\glo@parent
1440       {%
1441         \def\do@glshaschildren{#2}%
1442         \@endfortrue
1443       }%
1444     }%
1445   }%
1446   \do@glshaschildren
1447 }%
1448 }
```

`\ifglshasparent` `\ifglshasparent{<label>}{<true part>}{<false part>}`

```

1449 \newcommand{\ifglshasparent}[3]{%
1450   \glsdoifexists{#1}%
1451   {%
1452     \ifcsemt{glo@\glsdetoklabel{#1}@parent}{#3}{#2}%
1453   }%
1454 }

```

`\ifglshasdesc` `\ifglshasdesc{<label>}{<true part>}{<false part>}`

```

1455 \newcommand*{\ifglshasdesc}[3]{%
1456   \ifcsemt{glo@\glsdetoklabel{#1}@desc}{#3}%
1457   {%
1458     {#2}%
1459   }%

```

`sdescsuppressed` `\ifglsdscsuppressed{<label>}{<true part>}{<false part>}` Does *<true part>* if the description is just `\nopostdesc` otherwise does *<false part>*.

```

1460 \newcommand*{\ifglsdscsuppressed}[3]{%
1461   \ifcsemt{glo@\glsdetoklabel{#1}@desc}{@no@post@desc}%
1462   {%
1463     {#3}%
1464   }%

```

`\ifglshassymbol` `\ifglshassymbol{<label>}{<true part>}{<false part>}`

```

1465 \newcommand*{\ifglshassymbol}[3]{%
1466   \letcs{\@glo@symbol}{glo@\glsdetoklabel{#1}@symbol}%
1467   \ifdefempty\@glo@symbol
1468   {%
1469     {%
1470       \ifdefequal\@glo@symbol\@gls@default@value
1471       {%
1472         {#2}%
1473       }%
1474   }%

```

`\ifglshaslong` `\ifglshaslong{<label>}{<true part>}{<false part>}`

```

1475 \newcommand*{\ifglshaslong}[3]{%
1476   \letcs{\@glo@long}{glo@\glsdetoklabel{#1}@long}%
1477   \ifdefempty\@glo@long
1478   {%
1479     {%
1480       \ifdefequal\@glo@long\@gls@default@value
1481       {%
1482         {#2}%
1483       }%
1484   }%

```

```
\ifglshasshort \ifglshasshort{<label>}{<true part>}{<false part>}
1485 \newcommand*{\ifglshasshort}[3]{%
1486   \letcs{\@glo@short}{glo@glstdetoklabel{#1}@short}%
1487   \ifdefempty\@glo@short
1488     {#3}%
1489     {%
1490       \ifdefequal\@glo@short\@gls@default@value
1491         {#3}%
1492         {#2}%
1493     }%
1494 }
```

```
\ifglshasfield \ifglshasfield{<field>}{<label>}{<true part>}{<false part>}
```

```
1495 \newcommand*{\ifglshasfield}[4]{%
1496   \glstdoifexists{#2}%
1497   {%
1498     \letcs{\@glo@thisvalue}{glo@glstdetoklabel{#2}@#1}%
1499     \ifdef\@glo@thisvalue
1500     {%
1501       \ifdefempty\@glo@thisvalue
1502       {%
1503         #4%
1504       }%
1505       {%
1506         \ifdefequal\@glo@thisvalue\@gls@default@value
1507         {%
1508           #4%
1509         }%
1510         {%
1511           \let\glscurrentfieldvalue\@glo@thisvalue
1512           #3%
1513         }%
1514       }%
1515     }%
1516   }
```

Field given isn't defined, so check if mapping exists.

```
1517 \gls@fetchfield{\gls@thisfield}{#1}%
```

If `\gls@thisfield` is defined, we've found a map. If not, the field supplied doesn't exist.

```
1518 \ifdef\gls@thisfield
1519 {%
```

Is defined, so now check if empty.

```
1520 \letcs{\glo@thisvalue}{glo\glsdetoklabel{#2}@\gls@thisfield}%
1521 \ifdefempty\glo@thisvalue
1522 {%
```

Is empty so field hasn't been set.

```
1523 #4%
1524 }%
1525 {%
```

Isn't empty so check if it's been set to `\gls@default@value`.

```
1526 \ifdefequal\glo@thisvalue\gls@default@value
1527 {%
```

Value is set to the default value.

```
1528 #4%
1529 }%
1530 {%
```

Non-empty, non-default value. Allow user to access this value through `\glscurrentfieldvalue`.

```
1531 \let\glscurrentfieldvalue\glo@thisvalue
1532 #3%
1533 }%
1534 }%
1535 }%
1536 {%
```

Not defined.

```
1537 \GlossariesWarning{Unknown entry field '#1'}%
1538 #4%
1539 }%
1540 }%
1541 }%
1542 }
```

`\glscurrentfieldvalue`

```
1543 \newcommand*{\glscurrentfieldvalue}{}
```

1.6 Defining new glossaries

A comma-separated list of glossary names is stored in `\glo@types`. When a new glossary type is created, its identifying name is added to this list. This is used by commands that iterate through all glossaries (such as `\makeglossaries` and `\printglossaries`).

```

\@glo@types
1544 \newcommand*{\@glo@types}{,}

ide@newglossary If the user removes the glossary package from their document, ensure the next run doesn't
throw a load of undefined control sequence errors when the aux file is parsed.

1545 \newcommand*\@gls@provide@newglossary{%
1546   \protected@write\@auxout{}\string\providecommand\string\@newglossary[4]{}}%

Only need to do this once.

1547 \let\@gls@provide@newglossary\relax
1548 }

\defglsentryfmt Allow different glossaries to have different display styles.

1549 \newcommand*\defglsentryfmt}[2][\glsdefaultttype]{%
1550   \csgdef{gls@#1@entryfmt}{#2}%
1551 }

\gls@doentryfmt
1552 \newcommand*\gls@doentryfmt}[1]{\csuse{gls@#1@entryfmt}}

ls@forbidtextext As a security precaution, don't allow the user to specify a 'tex' extension for any of the glossary
files. (Just in case a seriously confused novice user doesn't know what they're doing.) The
argument must be a control sequence whose replacement text is the requested extension.

1553 \newcommand*\@gls@forbidtextext}[1]{%
1554   \ifboolexpr{test {\ifdefstring{#1}{tex}}
1555             or test {\ifdefstring{#1}{TEX}}}
1556   {%
1557     \def#1{nottex}%
1558     \PackageError{glossaries}%
1559       {Forbidden '.tex' extension replaced with '.nottex'}%
1560     {I'm sorry, I can't allow you to do something so reckless.\MessageBreak
1561       Don't use '.tex' as an extension for a temporary file.}%
1562   }%
1563   {%
1564   }%
1565 }

\gls@gobbleopt Discard optional argument.

1566 \newcommand*\gls@gobbleopt{\new@ifnextchar[\@gls@gobbleopt]}
1567 \def\@gls@gobbleopt[#1]{}

```

A new glossary type is defined using `\newglossary`. Syntax:

```
\newglossary[⟨log-ext⟩]{⟨name⟩}{⟨in-ext⟩}{⟨out-ext⟩} {⟨title⟩}[⟨counter⟩]
```

where `⟨log-ext⟩` is the extension of the makeindex transcript file, `⟨in-ext⟩` is the extension of the glossary input file (read in by `\printglossary` and created by makeindex), `⟨out-ext⟩`

is the extension of the glossary output file which is read in by `makeindex` (lines are written to this file by the `\glossary` command), *<title>* is the title of the glossary that is used in `\glossarysection` and *<counter>* is the default counter to be used by entries belonging to this glossary. The `makeglossaries` Perl script reads in the relevant extensions from the auxiliary file, and passes the appropriate file names and switches to `makeindex`.

`\newglossary`

```
1568 \newcommand*{\newglossary}{\@ifstar\s@newglossary\ns@newglossary}
```

`\s@newglossary` The starred version will construct the extension based on the label.

```
1569 \newcommand*{\s@newglossary}[2]{%
```

```
1570 \ns@newglossary[#1-glg]{#1}{#1-gls}{#1-glo}{#2}%
```

```
1571 }
```

`\ns@newglossary` Define the unstarred version.

```
1572 \newcommand*{\ns@newglossary}[5][glg]{%
```

```
1573 \doifglossarynoexistsordo{#2}%
```

```
1574 {%
```

Check if default has been set

```
1575 \ifundef\glsdefaultttype
```

```
1576 {%
```

```
1577 \gdef\glsdefaultttype{#2}%
```

```
1578 }{}%
```

Add this to the list of glossary types:

```
1579 \toks@{#2}\edef\@glo@types{\@glo@types\the\toks@,}%
```

Define a comma-separated list of labels for this glossary type, so that all the entries for this glossary can be reset with a single command. When a new entry is created, its label is added to this list.

```
1580 \expandafter\gdef\csname glolist@#2\endcsname{,}%
```

Store the file extensions:

```
1581 \expandafter\edef\csname @glotype@#2@log\endcsname{#1}%
```

```
1582 \expandafter\edef\csname @glotype@#2@in\endcsname{#3}%
```

```
1583 \expandafter\edef\csname @glotype@#2@out\endcsname{#4}%
```

```
1584 \expandafter\@gls@forbidtexext\csname @glotype@#2@log\endcsname
```

```
1585 \expandafter\@gls@forbidtexext\csname @glotype@#2@in\endcsname
```

```
1586 \expandafter\@gls@forbidtexext\csname @glotype@#2@out\endcsname
```

Store the title:

```
1587 \expandafter\def\csname @glotype@#2@title\endcsname{#5}%
```

```
1588 \@gls@provide@newglossary
```

```
1589 \protected@write\auxout{}{\string\@newglossary{#2}{#1}{#3}{#4}}%
```

How to display this entry in the document text (uses `\glsentry` by default). This can be re-defined by the user later if required (see `\defglsentry`). This may already have been defined if this has been specified as a list of acronyms.

```

1590 \ifcsundef{gls@#2@entryfmt}%
1591 {%
1592   \defglsentryfmt[#2]{\glsentryfmt}%
1593 }%
1594 {}%

```

Define sort counter if required:

```

1595 \@gls@defsortcount{#2}%

```

Find out if the final optional argument has been specified, and use it to set the counter associated with this glossary. (Uses `\glscounter` if no optional argument is present.)

```

1596 \@ifnextchar[{\@gls@setcounter{#2}}%
1597   {\@gls@setcounter{#2}[\glscounter]}%
1598 }%
1599 {%
1600   \gls@gobbleopt
1601 }%
1602 }

```

`\altnewglossary`

```

1603 \newcommand*{\altnewglossary}[3]{%
1604   \newglossary[#2-glg]{#1}{#2-gls}{#2-glo}{#3}%
1605 }

```

Only define new glossaries in the preamble:

```

1606 \@onlypreamble{\newglossary}

```

Only define new glossaries before `\makeglossaries`

```

1607 \@onlypremakeg\newglossary

```

`\@newglossary` is used to specify the file extensions for the `makeindex` input, output and transcript files. It is written to the auxiliary file by `\newglossary`. Since it is not used by \LaTeX , `\@newglossary` simply ignores its arguments.

`\@newglossary`

```

1608 \newcommand*{\@newglossary}[4]{}

```

Store counter to be used for given glossary type (the first argument is the glossary label, the second argument is the name of the counter):

`@gls@setcounter`

```

1609 \def\@gls@setcounter#1[#2]{%
1610   \expandafter\def\csname @glsotype@#1@counter\endcsname{#2}%

```

Add counter to xindy list, if not already added:

```

1611   \ifglsxindy
1612     \GlsAddXdyCounters{#2}%
1613   \fi
1614 }

```

Get counter associated with given glossary (the argument is the glossary label):

@gls@getcounter

```
1615 \newcommand*{\@gls@getcounter}[1]{%
1616 \csname @gls@#1@counter\endcsname
1617 }
```

Define the main glossary. This will be the first glossary to be displayed when using `\printglossaries`.

```
1618 \glsdefmain
```

Define the “acronym” glossaries if required.

```
1619 \@gls@do@acronymsdef
```

Define the “symbols”, “numbers” and “index” glossaries if required.

```
1620 \@gls@do@symbolsdef
```

```
1621 \@gls@do@numbersdef
```

```
1622 \@gls@do@indexdef
```

`ignoredglossary` Creates a new glossary that doesn’t have associated files. This glossary is ignored by and commands that iterate over glossaries, such as `\printglossaries`, and won’t work with commands like `\printglossary`. It’s intended for entries that are so commonly-known they don’t require a glossary.

```
1623 \newcommand*{\newignoredglossary}[1]{%
1624 \ifdefempty\@ignored@glossaries
1625 {%
1626 \edef\@ignored@glossaries{#1}%
1627 }%
1628 {%
1629 \eappto\@ignored@glossaries{,#1}%
1630 }%
1631 \csgdef{glolist@#1}{,}%
1632 \ifcsundef{gls@#1@entryfmt}%
1633 {%
1634 \defglsentryfmt[#1]{\glsentryfmt}%
1635 }%
1636 {}%
1637 \ifdefempty\@gls@nohyperlist
1638 {%
1639 \renewcommand*{\@gls@nohyperlist}{#1}%
1640 }%
1641 {%
1642 \eappto\@gls@nohyperlist{,#1}%
1643 }%
1644 }
```

`ored@glossaries` List of ignored glossaries.

```
1645 \newcommand*{\@ignored@glossaries}{}
```

`ignoredglossary` Tests if the given glossary is an ignored glossary. Expansion is used in case the first argument is a control sequence.

```

1646 \newcommand*{\ifignoredglossary}[3]{%
1647   \edef\@gls@igtype{#1}%
1648   \expandafter\DTLifinlist\expandafter
1649     {\@gls@igtype}{\@ignored@glossaries}{#2}{#3}%
1650 }

```

1.7 Defining new entries

New glossary entries are defined using `\newglossaryentry`. This command requires a label and a key-value list that defines the relevant information for that entry. The definition for these keys follows. Note that the name, description and symbol keys will be sanitized later, depending on the value of the package option `sanitize` (this means that if some of the keys haven't been defined, they can be constructed from the name and description key before they are sanitized).

name The name key indicates the name of the term being defined. This is how the term will appear in the glossary. The name key is required when defining a new glossary entry.

```

1651 \define@key{glossentry}{name}{%
1652 \def\@glo@name{#1}%
1653 }

```

description The description key is usually only used in the glossary, but can be made to appear in the text by redefining `\glsentryfmt` or using `\defglsentryfmt`. The description key is required when defining a new glossary entry. If a long description is required, use `\longnewglossaryentry` instead of `\newglossaryentry`.

```

1654 \define@key{glossentry}{description}{%
1655 \def\@glo@desc{#1}%
1656 }

```

descriptionplural

```

1657 \define@key{glossentry}{descriptionplural}{%
1658 \def\@glo@descplural{#1}%
1659 }

```

sort The sort key needs to be sanitized here (the sort key is provided for `makeindex`'s benefit, not for use in the document). The sort key is optional when defining a new glossary entry. If omitted, the value is given by *<name>* *<description>*.

```

1660 \define@key{glossentry}{sort}{%
1661 \def\@glo@sort{#1}}

```

text The text key determines how the term should appear when used in the document (i.e. outside of the glossary). If omitted, the value of the name key is used instead.

```

1662 \define@key{glossentry}{text}{%
1663 \def\@glo@text{#1}%
1664 }

```

plural The plural key determines how the plural form of the term should be displayed in the document. If omitted, the plural is constructed by appending `\glspluralsuffix` to the value of the text key.

```
1665 \define@key{glossentry}{plural}{%
1666 \def\@glo@plural{#1}%
1667 }
```

first The first key determines how the entry should be displayed in the document when it is first used. If omitted, it is taken to be the same as the value of the text key.

```
1668 \define@key{glossentry}{first}{%
1669 \def\@glo@first{#1}%
1670 }
```

firstplural The firstplural key is used to set the plural form for first use, in the event that the plural is required the first time the term is used. If omitted, it is constructed by appending `\glspluralsuffix` to the value of the first key.

```
1671 \define@key{glossentry}{firstplural}{%
1672 \def\@glo@firstplural{#1}%
1673 }
```

s@default@value

```
1674 \newcommand*{\@gls@default@value}{\relax}
```

symbol The symbol key is ignored by most of the predefined glossary styles, and defaults to `\relax` if omitted. It is provided for glossary styles that require an associated symbol, as well as a name and description. To make this value appear in the glossary, you need to redefine `\glossentry`. If you want this value to appear in the text when the term is used by commands like `\gls`, you will need to change `\glsentryfmt` (or use for `\defglsentryfmt` individual glossaries).

```
1675 \define@key{glossentry}{symbol}{%
1676 \def\@glo@symbol{#1}%
1677 }
```

symbolplural

```
1678 \define@key{glossentry}{symbolplural}{%
1679 \def\@glo@symbolplural{#1}%
1680 }
```

type The type key specifies to which glossary this entry belongs. If omitted, the default glossary is used.

```
1681 \define@key{glossentry}{type}{%
1682 \def\@glo@type{#1}}
```

counter The counter key specifies the name of the counter associated with this glossary entry:

```
1683 \define@key{glossentry}{counter}{%
1684 \ifcsundef{c@#1}%
```

```

1685 {%
1686   \PackageError{glossaries}%
1687   {There is no counter called ‘#1’}%
1688   {%
1689     The counter key should have the name of a valid counter
1690     as its value%
1691   }%
1692 }%
1693 {%
1694   \def\@glo@counter{#1}%
1695 }%
1696 }

```

see The see key specifies a list of cross-references

```

1697 \define@key{glossentry}{see}{%
1698   \gls@checkseeallowed
1699   \def\@glo@see{#1}%
1700   \@glo@seeautonumberlist
1701 }

```

checkseeallowed

```

1702 \newcommand*{\gls@checkseeallowed}{%
1703   \@gls@see@noindex
1704 }

```

ed@preambleonly

```

1705 \newcommand*{\gls@checkseeallowed@preambleonly}{%
1706   \GlossariesWarning{glossaries}%
1707   {‘see’ key doesn’t have any effect when used in the document
1708     environment. Move the definition to the preamble
1709     after \string\makeglossaries\space
1710     or \string\makenoidxglossaries}%
1711 }

```

parent The parent key specifies the parent entry, if required.

```

1712 \define@key{glossentry}{parent}{%
1713   \def\@glo@parent{#1}}

```

nonumberlist The nonumberlist key suppresses or activates the number list for the given entry.

```

1714 \define@choicekey{glossentry}{nonumberlist}[\val\nr]{true,false}[true]{%
1715   \ifcase\nr\relax
1716     \def\@glo@prefix{\glsnonextpages}%
1717     \@gls@savenonumberlist{true}%
1718   \else
1719     \def\@glo@prefix{\glsnextpages}%
1720     \@gls@savenonumberlist{false}%
1721   \fi
1722 }

```

savenonumberlist The nonumberlist option isn't saved by default (as it just sets the prefix) which isn't a problem when the entries are defined in the preamble, but causes a problem when entries are defined in the document. In this case, the value needs to be saved so that it can be written to the .glsdefs file.

```
1723 \newcommand*{\@gls@savenonumberlist}[1]{}
```

initnonumberlist

```
1724 \newcommand*{\@gls@initnonumberlist}{}%
```

nitnonumberlist

```
1725 \newcommand*{\@gls@storenonumberlist}[1]{}
```

savenonumberlist Allow the nonumberlist value to be saved.

```
1726 \newcommand*{\@gls@enablesavenonumberlist}{%
1727   \renewcommand*{\@gls@initnonumberlist}{%
1728     \undef\@glo@nonumberlist
1729   }%
1730   \renewcommand*{\@gls@savenonumberlist}[1]{%
1731     \def\@glo@nonumberlist{##1}%
1732   }%
1733   \renewcommand*{\@gls@storenonumberlist}[1]{%
1734     \ifdef\@glo@nonumberlist
1735       {%
1736         \cslet{glo@glsdetoklabel{##1}@nonumberlist}{\@glo@nonumberlist}%
1737       }%
1738     }%
1739   }%
1740   \appto\@gls@keymap{,{nonumberlist}{nonumberlist}}%
1741 }
```

Define some generic user keys. (Additional keys can be added by the user.)

user1

```
1742 \define@key{glossentry}{user1}{%
1743   \def\@glo@useri{#1}%
1744 }
```

user2

```
1745 \define@key{glossentry}{user2}{%
1746   \def\@glo@userii{#1}%
1747 }
```

user3

```
1748 \define@key{glossentry}{user3}{%
1749   \def\@glo@useriii{#1}%
1750 }
```

user4

```
1751 \define@key{glossentry}{user4}{%
1752   \def\@glo@useriv{#1}%
1753 }
```

user5

```
1754 \define@key{glossentry}{user5}{%
1755   \def\@glo@userv{#1}%
1756 }
```

user6

```
1757 \define@key{glossentry}{user6}{%
1758   \def\@glo@uservi{#1}%
1759 }
```

short This key is provided for use by \newacronym. It's not designed for general purpose use, so isn't described in the user manual.

```
1760 \define@key{glossentry}{short}{%
1761   \def\@glo@short{#1}%
1762 }
```

shortplural This key is provided for use by \newacronym.

```
1763 \define@key{glossentry}{shortplural}{%
1764   \def\@glo@shortpl{#1}%
1765 }
```

long This key is provided for use by \newacronym.

```
1766 \define@key{glossentry}{long}{%
1767   \def\@glo@long{#1}%
1768 }
```

longplural This key is provided for use by \newacronym.

```
1769 \define@key{glossentry}{longplural}{%
1770   \def\@glo@longpl{#1}%
1771 }
```

\@glsnname Define command to generate error if name key is missing.

```
1772 \newcommand*\@glsnname{%
1773   \PackageError{glossaries}{name key required in
1774   \string\newglossaryentry\space for entry '\@glo@label'}{You
1775   haven't specified the entry name}}
```

\@glsnodelsc Define command to generate error if description key is missing.

```
1776 \newcommand*\@glsnodelsc{%
1777   \PackageError{glossaries}
1778   {%
1779     description key required in \string\newglossaryentry\space
1780     for entry '\@glo@label'%
1781   }}
```



```

1781 }%
1782 {%
1783     You haven't specified the entry description%
1784 }%
1785 }%

\lsdefaultplural Now obsolete. Don't use.
1786 \newcommand*{\@glsdefaultplural}{}

\issingnumberlist Define a command to generate warning when numberlist not set.
1787 \newcommand*{\@gls@missingnumberlist}[1]{%
1788     ??%
1789     \ifglssavenumberlist
1790         \GlossariesWarning{Missing number list for entry '#1'.
1791             Maybe makeglossaries + rerun required}%
1792     \else
1793         \PackageError{glossaries}%
1794             {Package option 'savenumberlist=true' required}%
1795         {%
1796             You must use the 'savenumberlist' package option
1797             to reference location lists.%
1798         }%
1799     \fi
1800 }

@glsdefaultsort Define command to set default sort.
1801 \newcommand*{\@glsdefaultsort}{\@glo@name}

\gls@level Register to increment entry levels.
1802 \newcount\gls@level

@noexpand@field
1803 \newcommand{\@gls@noexpand@field}[3]{%
1804     \expandafter\global\expandafter
1805         \let\csname glo@#1@#2\endcsname#3%
1806 }

noexpand@fields
1807 \newcommand{\@gls@noexpand@fields}[4]{%
1808     \ifcsdef{gls@assign@#3@field}
1809     {%
1810         \ifdefequal{#4}{\@gls@default@value}%
1811         {%
1812             \edef\@gls@value{\expandonce{#1}}%
1813             \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
1814         }%
1815     }%
1816     \csuse{gls@assign@#3@field}{#2}{#4}%

```

```

1817     }%
1818 }%
1819 {%
1820   \ifdefequal{#4}{\@gls@default@value}%
1821   {%
1822     \edef\@gls@value{\expandonce{#1}}%
1823     \@gls@noexpand@field{#2}{#3}{\@gls@value}%
1824   }%
1825   {%
1826     \@gls@noexpand@field{#2}{#3}{#4}%
1827   }%
1828 }%
1829 }

```

ls@expand@field

```

1830 \newcommand{\@gls@expand@field}[3]{%
1831   \expandafter
1832   \protected@xdef\csname glo@#1@#2\endcsname{#3}%
1833 }

```

s@expand@fields

```

1834 \newcommand{\@gls@expand@fields}[4]{%
1835   \ifcsdef{gls@assign@#3@field}
1836   {%
1837     \ifdefequal{#4}{\@gls@default@value}%
1838     {%
1839       \edef\@gls@value{\expandonce{#1}}%
1840       \csuse{gls@assign@#3@field}{#2}{\@gls@value}%
1841     }%
1842     {%
1843       \expandafter\@gls@startswitexpandonce#4\relax\relax\gls@endcheck
1844       {%
1845         \@gls@expand@field{#2}{#3}{#4}%
1846       }%
1847       {%
1848         \csuse{gls@assign@#3@field}{#2}{#4}%
1849       }%
1850     }%
1851   }%
1852   {%
1853     \ifdefequal{#4}{\@gls@default@value}%
1854     {%
1855       \@gls@expand@field{#2}{#3}{#1}%
1856     }%
1857     {%
1858       \@gls@expand@field{#2}{#3}{#4}%
1859     }%
1860   }%
1861 }

```

swithexpandonce

```
1862 \def\@gls@expandonce{\expandonce}
1863 \def\@gls@startswithexpandonce#1#2\gls@endcheck#3#4{%
1864   \def\@gls@tmp{#1}%
1865   \ifdefequal{\@gls@expandonce}{\@gls@tmp}{#3}{#4}%
1866 }
```

gls@assign@field

```
\gls@assign@field{\<def value>}{\<label>}{\<field>}{\<tmp cs>}
```

Assigns an entry field. Expansion performed by default (except for name, symbol and description where backward compatibility required). If *<tmp cs>* is *<@gls@default@value>*, *<def value>* is used instead.

```
1867 \let\gls@assign@field\@gls@expand@fields
```

glsexpandfields

Fully expand values when assigning fields (except for specific fields that are overridden by *\glssetnoexpandfield*).

```
1868 \newcommand*{\glsexpandfields}{%
1869   \let\gls@assign@field\@gls@expand@fields
1870 }
```

snoexpandfields

Don't expand values when assigning fields (except for specific fields that are overridden by *\glssetexpandfield*).

```
1871 \newcommand*{\glsnoexpandfields}{%
1872   \let\gls@assign@field\@gls@noexpand@fields
1873 }
```

ewglossaryentry

Define *\newglossaryentry* *{\<label>}* *{\<key-val list>}*. There are two required fields in *<key-val list>*: name (or parent) and description. (See above.)

```
1874 \newrobustcmd{\newglossaryentry}[2]{%
```

Check to see if this glossary entry has already been defined:

```
1875   \glsdoifnoexists{#1}%
1876   {%
1877     \gls@defglossaryentry{#1}{#2}%
1878   }%
1879 }
```

ewglossaryentry

The definition of *\newglossaryentry* is changed at the start of the document environment. The see key doesn't work for entries that have been defined in the document environment.

```
1880 \newcommand*{\gls@defdocnewglossaryentry}{%
1881   \let\gls@checkseeallowed\gls@checkseeallowed@preambleonly
1882   \let\newglossaryentry\new@glossaryentry
1883 }
```

deglossaryentry

Like *\newglossaryentry* but does nothing if the entry has already been defined.

```
1884 \newrobustcmd{\provideglossaryentry}[2]{%
```

```

1885 \ifglentryexists{#1}%
1886 {}%
1887 {%
1888   \gls@defglossaryentry{#1}{#2}%
1889 }%
1890 }
1891 \onlypreamble{\provideglossaryentry}

```

`w@glossaryentry` For use in document environment.

```

1892 \newrobustcmd{\new@glossaryentry}[2]{%
1893   \ifundef\@gls@deffile
1894   {%
1895     \global\newwrite\@gls@deffile
1896     \immediate\openout\@gls@deffile=\jobname.glsdefs
1897   }%
1898   {}%
1899   \ifglentryexists{#1}{}%
1900   {%
1901     \gls@defglossaryentry{#1}{#2}%
1902   }%
1903   \@gls@writedef{#1}%
1904 }
1905 \AtBeginDocument
1906 {
1907   \@gls@enablesavenonumberlist
1908   \makeatletter
1909   \InputIfFileExists{\jobname.glsdefs}{-}{-}%
1910   \makeatother
1911   \gls@defdocnewglossaryentry
1912 }
1913 \AtEndDocument{\ifdef\@gls@deffile{\closeout\@gls@deffile}{-}}

```

`\@gls@writedef` Writes glossary entry definition to `\@gls@deffile`.

```

1914 \newcommand*{\@gls@writedef}[1]{%
1915   \immediate\write\@gls@deffile
1916   {%
1917     \string\ifglentryexists{#1}{}\glspercentchar^^J%
1918     \expandafter\@gobble\string\{\glspercentchar^^J%
1919     \string\gls@defglossaryentry{\glsdetoklabel{#1}}\glspercentchar^^J%
1920     \expandafter\@gobble\string\{\glspercentchar%
1921   }%

```

Write key value information:

```

1922 \@for\@gls@map:=\@gls@keymap\do
1923 {%
1924   \letcs\glo@value{glo@\glsdetoklabel{#1}}\expandafter\@secondoftwo\@gls@map}%
1925   \ifdef\glo@value
1926   {%
1927     \@onelevel@sanitize\glo@value
1928     \immediate\write\@gls@deffile

```

```

1929      {%
1930          \expandafter\@firstoftwo\@gls@map
1931          =\expandafter\@gobble\string\{\@glo@value\expandafter\@gobble\string\},%
1932          \glspercentchar
1933      }%
1934  }%
1935  {}%
1936  }%

```

Provide hook:

```

1937  \gls.writedefhook
1938  \immediate\write\@gls@deffile
1939  {%
1940      \glspercentchar^^J%
1941      \expandafter\@gobble\string\}\glspercentchar^^J%
1942      \expandafter\@gobble\string\}\glspercentchar%
1943  }%
1944  }

```

`\@gls@keymap` List of entry definition key names and corresponding tag in control sequence used to store the value.

```

1945 \newcommand*{\@gls@keymap}{%
1946   {name}{name},%
1947   {sort}{sortvalue},% unescaped sort value
1948   {type}{type},%
1949   {first}{first},%
1950   {firstplural}{firstpl},%
1951   {text}{text},%
1952   {plural}{plural},%
1953   {description}{desc},%
1954   {descriptionplural}{descplural},%
1955   {symbol}{symbol},%
1956   {symbolplural}{symbolplural},%
1957   {user1}{useri},%
1958   {user2}{userii},%
1959   {user3}{useriii},%
1960   {user4}{useriv},%
1961   {user5}{userv},%
1962   {user6}{uservi},%
1963   {long}{long},%
1964   {longplural}{longpl},%
1965   {short}{short},%
1966   {shortplural}{shortpl},%
1967   {counter}{counter},%
1968   {parent}{parent}%
1969 }

```

`\@gls@fetchfield` `\@gls@fetchfield{<cs>}{<field>}`

Fetches the internal field label from the given user *<field>* and stores in *<cs>*.

```
1970 \newcommand*{\@gls@fetchfield}[2]{%
```

Ensure user field name is fully expanded

```
1971 \edef\@gls@thisval{#2}%
```

Iterate through known mappings until we find the one for this field.

```
1972 \@for\@gls@map:=\@gls@keymap\do{%
```

```
1973 \edef\@this@key{\expandafter\@firstoftwo\@gls@map}%
```

```
1974 \ifdefequal{\@this@key}{\@gls@thisval}%
```

```
1975 {%
```

Found it.

```
1976 \edef#1{\expandafter\@secondoftwo\@gls@map}%
```

Break out of loop.

```
1977 \@endfortrue
```

```
1978 }%
```

```
1979 {}%
```

```
1980 }%
```

```
1981 }
```

glsaddstoragekey

```
\glsaddstoragekey{<key>}{<default value>}{<no link cs>}
```

Similar to `\glsaddkey` but intended for keys whose values aren't explicitly used in the document, but might be required behind the scenes by other commands.

```
1982 \newcommand*{\glsaddstoragekey}{\@ifstar\@sglsaddstoragekey\@glsaddstoragekey}
```

Starred version switches on expansion for this key.

```
1983 \newcommand*{\@sglsaddstoragekey}[1]{%
```

```
1984 \key@ifundefined{glossentry}{#1}%
```

```
1985 {%
```

```
1986 \expandafter\newcommand\expandafter*\expandafter
```

```
1987 {\csname gls@assign@#1@field\endcsname}[2]{%
```

```
1988 \@gls@expand@field{##1}{#1}{##2}%
```

```
1989 }%
```

```
1990 }%
```

```
1991 {}%
```

```
1992 \@glsaddstoragekey{#1}%
```

```
1993 }
```

Unstarred version doesn't override default expansion.

```
1994 \newcommand*{\@glsaddstoragekey}[3]{%
```

Check the specified key doesn't already exist.

```
1995 \key@ifundefined{glossentry}{#1}%
```

```
1996 {%
```

Set up the key.

```
1997 \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
```

```
1998 \appto\@gls@keymap{, {#1}{#1}}%
```

Set the default value.

```
1999 \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
```

Assignment code.

```
2000 \appto\@newglossaryentryposthook{%
2001 \letcs{\@glo@tmp}{@glo@#1}%
2002 \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
2003 }%
```

Define the no-link commands.

```
2004 \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
2005 }%
2006 {%
2007 \PackageError{glossaries}{Key ‘#1’ already exists}{}%
2008 }%
2009 }
```

```
\glsaddkey \glsaddkey{<key>}{<default value>}{<no link cs>}{<no link ucfirst cs>}
           {\<link cs>}{\<link ucfirst cs>}{\<link allcaps cs>}
```

Allow user to add their own custom keys.

```
2010 \newcommand*{\glsaddkey}{\ifstar\@sglsaddkey\@glsaddkey}
```

Starred version switches on expansion for this key.

```
2011 \newcommand*{\@sglsaddkey}[1]{%
2012 \key@ifundefined{glossentry}{#1}%
2013 {%
2014 \expandafter\newcommand\expandafter*\expandafter
2015 {\csname gls@assign@#1@field\endcsname}[2]{%
2016 \@gls@expand@field{##1}{#1}{##2}%
2017 }%
2018 }%
2019 }%
2020 \@glsaddkey{#1}%
2021 }
```

Unstarred version doesn't override default expansion.

```
2022 \newcommand*{\@glsaddkey}[7]{%
```

Check the specified key doesn't already exist.

```
2023 \key@ifundefined{glossentry}{#1}%
2024 {%
```

Set up the key.

```
2025 \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
2026 \appto\@gls@keymap{,}{#1}{#1}}%
```

Set the default value.

```
2027 \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
```

Assignment code.

```
2028 \appto\@newglossaryentryposthook{%
2029 \letcs{\@glo@tmp}{\@glo@#1}%
2030 \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
2031 }%
```

Define the no-link commands.

```
2032 \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
2033 \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%
```

Now for the commands with links. First the version with no case change:

```
2034 \ifcsdef{@gls@user@#1@}%
2035 {%
2036 \PackageError{glossaries}%
2037 {Can't define '\string#5' as helper command
2038 '\expandafter\string\csname @gls@user@#1@endcsname' already exists}%
2039 }%
2040 }%
2041 {%
2042 \expandafter\newcommand\expandafter*\expandafter
2043 {\csname @gls@user@#1@endcsname}[2][\%
2044 \new@ifnextchar[%
2045 {\csuse{@gls@user@#1@}{##1}{##2}}%
2046 {\csuse{@gls@user@#1@}{##1}{##2}[]}}%
2047 \csdef{@gls@user@#1@}##1##2[##3]{%
2048 \@gls@field@link{##1}{##2}{#3{##2}##3}%
2049 }%
2050 \newrobustcmd*{#5}{%
2051 \expandafter\@gls@hyp@opt\csname @gls@user@#1@endcsname}%
2052 }%
```

Next the version with the first letter converted to upper case:

```
2053 \ifcsdef{@Gls@user@#1@}%
2054 {%
2055 \PackageError{glossaries}%
2056 {Can't define '\string#6' as helper command
2057 '\expandafter\string\csname @Gls@user@#1@endcsname' already exists}%
2058 }%
2059 }%
2060 {%
2061 \expandafter\newcommand\expandafter*\expandafter
2062 {\csname @Gls@user@#1@endcsname}[2][\%
2063 \new@ifnextchar[%
2064 {\csuse{@Gls@user@#1@}{##1}{##2}}%
2065 {\csuse{@Gls@user@#1@}{##1}{##2}[]}}%
2066 \csdef{@Gls@user@#1@}##1##2[##3]{%
2067 \@gls@field@link{##1}{##2}{#4{##2}##3}%
2068 }%
2069 \newrobustcmd*{#6}{%
```



```

2070      \expandafter\@gls@hyp@opt\csname @Gls@user@#1\endcsname}%
2071  }%

  Finally the all caps version:
2072  \ifcsdef{@GLS@user@#1@}%
2073  {%
2074    \PackageError{glossaries}%
2075    {Can't define '\string#7' as helper command
2076    '\expandafter\string\csname @GLS@user@#1@\endcsname' already exists}%
2077    }%
2078  }%
2079  {%

2080  \expandafter\newcommand\expandafter*\expandafter
2081    {\csname @GLS@user@#1\endcsname}[2][{}]{%
2082    \new@ifnextchar[%
2083      {\csuse{@GLS@user@#1@}{##1}{##2}}%
2084      {\csuse{@GLS@user@#1@}{##1}{##2}[{}]}%
2085    \csdef{@GLS@user@#1@}##1##2[##3]{%
2086      \@gls@field@link{##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}%
2087    }%
2088    \newrobustcmd*{#7}{%
2089      \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
2090    }%
2091  }%
2092  {%
2093    \PackageError{glossaries}{Key '#1' already exists}{}%
2094  }%
2095 }

```

`\glsfieldxdef` `\glsfieldxdef{<label>}{<field>}{<definition>}`

```

2096 \newcommand{\glsfieldxdef}[3]{%
2097   \glsdoifexists{#1}%
2098   {%
2099     \edef\@glo@label{\glsdetoklabel{#1}}%
2100     \ifcsdef{glo@\@glo@label @#2}%
2101     {%
2102       \expandafter\xdef\csname glo@\@glo@label @#2\endcsname{#3}%
2103     }%
2104     {%
2105       \PackageError{glossaries}{Key '#2' doesn't exist}{}%
2106     }%
2107   }%
2108 }

```

`\glsfielddedef` `\glsfielddedef{<label>}{<field>}{<definition>}`

```

2109 \newcommand{\glsfielddedef}[3]{%
2110   \glsdoifexists{#1}%
2111   {%
2112     \edef\@glo@label{\glsdetoklabel{#1}}%
2113     \ifcsdef{glo@\@glo@label @#2}%
2114     {%
2115       \expandafter\edef\csname glo@\@glo@label @#2\endcsname{#3}%
2116     }%
2117     {%
2118       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2119     }%
2120   }%
2121 }

```

`\glsfieldgdef` `\glsfieldgdef{<label>}{<field>}{<definition>}`

```

2122 \newcommand{\glsfieldgdef}[3]{%
2123   \glsdoifexists{#1}%
2124   {%
2125     \edef\@glo@label{\glsdetoklabel{#1}}%
2126     \ifcsdef{glo@\@glo@label @#2}%
2127     {%
2128       \expandafter\gdef\csname glo@\@glo@label @#2\endcsname{#3}%
2129     }%
2130     {%
2131       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2132     }%
2133   }%
2134 }

```

`\glsfieldddef` `\glsfieldddef{<label>}{<field>}{<definition>}`

```

2135 \newcommand{\glsfieldddef}[3]{%
2136   \glsdoifexists{#1}%
2137   {%
2138     \edef\@glo@label{\glsdetoklabel{#1}}%
2139     \ifcsdef{glo@\@glo@label @#2}%
2140     {%
2141       \expandafter\def\csname glo@\@glo@label @#2\endcsname{#3}%
2142     }%
2143     {%
2144       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2145     }%
2146   }%

```

2147 }

`\glsfieldfetch` `\glsfieldfetch{<label>}{<field>}{<cs>}`

Fetches the value of the given field and stores in the given control sequence.

```
2148 \newcommand{\glsfieldfetch}[3]{%
2149   \glsdoifexists{#1}%
2150   {%
2151     \edef\@glo@label{\glsdetoklabel{#1}}%
2152     \ifcsdef{glo@\@glo@label @#2}%
2153     {%
2154       \letcs#3{glo@\@glo@label @#2}%
2155     }%
2156     {%
2157       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2158     }%
2159   }%
2160 }
```

`\ifglsfieldeq` `\ifglsfieldeq{<label>}{<field>}{<string>}{<true>}{<false>}`

Tests if the value of the given field is equal to the given string.

```
2161 \newcommand{\ifglsfieldeq}[5]{%
2162   \glsdoifexists{#1}%
2163   {%
2164     \edef\@glo@label{\glsdetoklabel{#1}}%
2165     \ifcsdef{glo@\@glo@label @#2}%
2166     {%
2167       \ifcsstring{glo@\@glo@label @#2}{#3}{#4}{#5}%
2168     }%
2169     {%
2170       \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2171     }%
2172   }%
2173 }
```

`\ifglsfielddefeq` `\ifglsfielddefeq{<label>}{<field>}{<command>}{<true>}{<false>}`

Tests if the value of the given field is equal to the replacement text of the given command.

```
2174 \newcommand{\ifglsfielddefeq}[5]{%
2175   \glsdoifexists{#1}%
2176   {%
2177     \edef\@glo@label{\glsdetoklabel{#1}}%
2178     \ifcsdef{glo@\@glo@label @#2}%
2179     {%
```

```

2180 \expandafter\ifdefstrequal
2181 \csname glo@\@glo@label @#2\endcsname{#3}{#4}{#5}%
2182 }%
2183 {%
2184 \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2185 }%
2186 }%
2187 }

```

```

\ifglsfieldcseq \ifglsfieldcseq{<label>}{<field>}{<cs name>}{<true>}{<false>}

```

As above but uses \ifcsstrequal instead of \ifdefstrequal

```

2188 \newcommand{\ifglsfieldcseq}[5]{%
2189 \glsdoifexists{#1}%
2190 {%
2191 \edef\@glo@label{\glsdetoklabel{#1}}%
2192 \ifcsdef{glo@\@glo@label @#2}%
2193 {%
2194 \ifcsstrequal{glo@\@glo@label @#2}{#3}{#4}{#5}%
2195 }%
2196 {%
2197 \PackageError{glossaries}{Key ‘#2’ doesn’t exist}{}%
2198 }%
2199 }%
2200 }

```

gls.writedefhook

```

2201 \newcommand*{\gls.writedefhook}{}

```

gls@assign@desc

```

2202 \newcommand*{\gls@assign@desc}[1]{%
2203 \gls@assign@field{#1}{desc}{\@glo@desc}%
2204 \gls@assign@field{\@glo@desc}{#1}{descplural}{\@glo@descplural}%
2205 }

```

ewglossaryentry

```

2206 \newcommand{\longnewglossaryentry}[3]{%
2207 \glsdoifnoexists{#1}%
2208 {%
2209 \bgroup
2210 \let\@org@newglossaryentryprehook\@newglossaryentryprehook
2211 \long\def\@newglossaryentryprehook{%
2212 \long\def\@glo@desc{#3\leavevmode\unskip\nopostdesc}%
2213 \@org@newglossaryentryprehook
2214 }%
2215 \renewcommand*{\gls@assign@desc}[1]{%
2216 \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%

```

```

2217         \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@desc}%
2218     }
2219     \gls@defglossaryentry{#1}{#2}%
2220 \egroup
2221 }
2222 }

```

Only allowed in the preamble. (Otherwise a long description could cause problems when writing the entry definition to the temporary file.)

```
2223 \@onlypreamble{\longnewglossaryentry}
```

`deglossaryentry` As the above but only defines the entry if it doesn't already exist.

```

2224 \newcommand{\longprovideglossaryentry}[3]{%
2225   \ifglstryexists{#1}{}%
2226   {\longnewglossaryentry{#1}{#2}{#3}}%
2227 }
2228 \@onlypreamble{\longprovideglossaryentry}

```

`defglossaryentry` `\gls@defglossaryentry{<label>}{<key-val list>}`

Defines a new entry without checking if it already exists.

```
2229 \newcommand{\gls@defglossaryentry}[2]{%
```

Prevent any further use of `\GlsSetQuote`:

```
2230   \let\GlsSetQuote\gls@nosetquote
```

Store label

```
2231   \edef\@glo@label{\glsdetoklabel{#1}}%
```

Provide a means for user defined keys to reference the label:

```
2232   \let\glslabel\@glo@label
```

Set up defaults. If the name or description keys are omitted, an error will be generated.

```
2233   \let\@glo@name\@glsnname
```

```
2234   \let\@glo@desc\@glsndesc
```

```
2235   \let\@glo@descplural\@gls@default@value
```

```
2236   \let\@glo@type\@gls@default@value
```

```
2237   \let\@glo@symbol\@gls@default@value
```

```
2238   \let\@glo@symbolplural\@gls@default@value
```

```
2239   \let\@glo@text\@gls@default@value
```

```
2240   \let\@glo@plural\@gls@default@value
```

Using `\let` instead of `\def` to make later comparison avoid expansion issues. (Thanks to Ulrich Diez for suggesting this.)

```
2241   \let\@glo@first\@gls@default@value
```

```
2242   \let\@glo@firstplural\@gls@default@value
```

Set the default sort:

```
2243 \let\@glo@sort\@gls@default@value
```

Set the default counter:

```
2244 \let\@glo@counter\@gls@default@value
```

```
2245 \def\@glo@see{}%
```

```
2246 \def\@glo@parent{}%
```

```
2247 \def\@glo@prefix{}%
```

Initialise nonnumberlist setting if we're in the document environment.

```
2248 \@gls@initnonnumberlist
```

```
2249 \def\@glo@useri{}%
```

```
2250 \def\@glo@userii{}%
```

```
2251 \def\@glo@useriii{}%
```

```
2252 \def\@glo@useriv{}%
```

```
2253 \def\@glo@userv{}%
```

```
2254 \def\@glo@uservi{}%
```

```
2255 \def\@glo@short{}%
```

```
2256 \def\@glo@shortpl{}%
```

```
2257 \def\@glo@long{}%
```

```
2258 \def\@glo@longpl{}%
```

Add start hook in case another package wants to add extra keys.

```
2259 \@newglossaryentryprehook
```

Extract key-val information from third parameter:

```
2260 \setkeys{glossentry}{#2}%
```

Check there is a default glossary.

```
2261 \ifundef\glsdefaultttype
```

```
2262 {%
```

```
2263 \PackageError{glossaries}%
```

```
2264 {No default glossary type (have you used 'nomain' by mistake?)}%
```

```
2265 {If you use package option 'nomain' you must define
```

```
2266 a new glossary before you can define entries}%
```

```
2267 }%
```

```
2268 {}}%
```

Assign type. This must be fully expandable

```
2269 \gls@assign@field{\glsdefaultttype}{\@glo@label}{type}{\@glo@type}%
```

```
2270 \edef\@glo@type{\glsentrytype{\@glo@label}}%
```

Check to see if this glossary type has been defined, if it has, add this label to the relevant list, otherwise generate an error.

```
2271 \ifcsundef{glolist@\@glo@type}%
```

```
2272 {%
```

```
2273 \PackageError{glossaries}%
```

```

2274      {Glossary type ‘\@glo@type’ has not been defined}%
2275      {You need to define a new glossary type, before making entries
2276      in it}%
2277  }%
2278  {%

  Check if it's an ignored glossary
2279      \ifignoredglossary\@glo@type
2280      {%

    The description may be omitted for an entry in an ignored glossary.
2281      \ifx\@glo@desc\@glsnodesc
2282      \let\@glo@desc\@empty
2283      \fi
2284  }%
2285  {%
2286  }%
2287      \protected@edef\@glo@list@{\csname glo@list@\@glo@type\endcsname}%
2288      \expandafter\edef\csname glo@list@\@glo@type\endcsname{%
2289      \@glo@list@{\@glo@label},}%
2290  }%

  Initialise level to 0.
2291      \gls@level=0\relax

  Has this entry been assigned a parent?
2292      \ifx\@glo@parent\@empty

    Doesn't have a parent. Set \glo@<label>@parent to empty.
2293      \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2294      \else

    Has a parent. Check to ensure this entry isn't its own parent.
2295      \ifdefequal\@glo@label\@glo@parent%
2296      {%
2297          \PackageError{glossaries}{Entry ‘\@glo@label’ can't be its own parent}{}%
2298          \def\@glo@parent{}%
2299          \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2300      }%
2301      {%

    Check the parent exists:
2302      \ifglsentryexists{\@glo@parent}%
2303      {%

    Parent exists. Set \glo@<label>@parent.
2304      \expandafter\edef\csname glo@\@glo@label @parent\endcsname{%
2305      \@glo@parent}%

    Determine level.
2306      \gls@level=\csname glo@\@glo@parent @level\endcsname\relax
2307      \advance\gls@level by 1\relax

```

If name hasn't been specified, use same as the parent name

```
2308      \ifx\@glo@name\@glsnoname
2309      \expandafter\let\expandafter\@glo@name
2310      \csname glo@\@glo@parent @name\endcsname
```

If name and plural haven't been specified, use same as the parent

```
2311      \ifx\@glo@plural\@gls@default@value
2312      \expandafter\let\expandafter\@glo@plural
2313      \csname glo@\@glo@parent @plural\endcsname
2314      \fi
2315      \fi
2316      }%
2317      {%
```

Parent doesn't exist, so issue an error message and change this entry to have no parent

```
2318      \PackageError{glossaries}%
2319      {%
2320      Invalid parent '\@glo@parent'
2321      for entry '\@glo@label' - parent doesn't exist%
2322      }%
2323      {%
2324      Parent entries must be defined before their children%
2325      }%
2326      \def\@glo@parent{}%
2327      \expandafter\gdef\csname glo@\@glo@label @parent\endcsname{}%
2328      }%
2329      }%
2330      \fi
```

Set the level for this entry

```
2331      \expandafter\xdef\csname glo@\@glo@label @level\endcsname{\number\gls@level}%
```

Define commands associated with this entry:

```
2332      \gls@assign@field{\@glo@name}{\@glo@label}{sortvalue}{\@glo@sort}%
2333      \letcs\@glo@sort{glo@\@glo@label @sortvalue}%
2334      \gls@assign@field{\@glo@name}{\@glo@label}{text}{\@glo@text}%
2335      \expandafter\gls@assign@field\expandafter
2336      {\csname glo@\@glo@label @text\endcsname\glspluralsuffix}%
2337      {\@glo@label}{plural}{\@glo@plural}%
2338      \expandafter\gls@assign@field\expandafter
2339      {\csname glo@\@glo@label @text\endcsname}%
2340      {\@glo@label}{first}{\@glo@first}%
```

If first has been specified, make the default by appending \glspluralsuffix, otherwise make the default the value of the plural key.

```
2341      \ifx\@glo@first\@gls@default@value
2342      \expandafter\gls@assign@field\expandafter
2343      {\csname glo@\@glo@label @plural\endcsname}%
2344      {\@glo@label}{firstpl}{\@glo@firstplural}%
2345      \else
2346      \expandafter\gls@assign@field\expandafter
```



```

2347         {\csname glo@\@glo@label @first\endcsname\glspluralsuffix}%
2348         {\@glo@label}{firstpl}{\@glo@firstplural}%
2349     \fi

2350     \ifcsundef{@glo@type@\@glo@type @counter}%
2351     {%
2352         \def\@glo@defaultcounter{\glscounter}%
2353     }%
2354     {%
2355         \letcs\@glo@defaultcounter{@glo@type@\@glo@type @counter}%
2356     }%
2357     \gls@assign@field{\@glo@defaultcounter}{\@glo@label}{counter}{\@glo@counter}%
2358     \gls@assign@field{}{\@glo@label}{useri}{\@glo@useri}%
2359     \gls@assign@field{}{\@glo@label}{userii}{\@glo@userii}%
2360     \gls@assign@field{}{\@glo@label}{useriii}{\@glo@useriii}%
2361     \gls@assign@field{}{\@glo@label}{useriv}{\@glo@useriv}%
2362     \gls@assign@field{}{\@glo@label}{userv}{\@glo@userv}%
2363     \gls@assign@field{}{\@glo@label}{uservi}{\@glo@uservi}%
2364     \gls@assign@field{}{\@glo@label}{short}{\@glo@short}%
2365     \gls@assign@field{}{\@glo@label}{shortpl}{\@glo@shortpl}%
2366     \gls@assign@field{}{\@glo@label}{long}{\@glo@long}%
2367     \gls@assign@field{}{\@glo@label}{longpl}{\@glo@longpl}%
2368     \ifx\@glo@name\@glsnoname
2369         \@glsnoname
2370         \let\@glo@name\@gls@default@value
2371     \fi
2372     \gls@assign@field{}{\@glo@label}{name}{\@glo@name}%

```

Set default numberlist if not defined:

```

2373     \ifcsundef{glo@\@glo@label @numberlist}%
2374     {%
2375         \csxdef{glo@\@glo@label @numberlist}{%
2376             \noexpand\@gls@missingnumberlist{\@glo@label}}%
2377     }%
2378     {}%

```

Store nonnumberlist setting if we're in the document environment.

```

2379     \@gls@storenonumberlist{\@glo@label}%

```

The smaller and smallcaps options set the description to \@glo@first. Need to check for this, otherwise it won't get expanded if the description gets sanitized.

```

2380     \def\@glo@@desc{\@glo@first}%
2381     \ifx\@glo@desc\@glo@@desc
2382         \let\@glo@desc\@glo@first
2383     \fi
2384     \ifx\@glo@desc\@glsnodesc
2385         \@glsnodesc
2386         \let\@glo@desc\@gls@default@value
2387     \fi
2388     \gls@assign@desc{\@glo@label}%

```

Set the sort key for this entry:

```
2389 \gls@defsort{\@glo@type}{\@glo@label}%
2390 \def\@glo@@symbol{\@glo@text}%
2391 \ifx\@glo@symbol\@glo@@symbol
2392 \let\@glo@symbol\@glo@text
2393 \fi
2394 \gls@assign@field{\relax}{\@glo@label}{symbol}{\@glo@symbol}%
2395 \expandafter
2396 \gls@assign@field\expandafter
2397 {\csname glo@\@glo@label @symbol\endcsname}
2398 {\@glo@label}{symbolplural}{\@glo@symbolplural}%
```

Define an associated boolean variable to determine whether this entry has been used yet (needs to be defined globally):

```
2399 \expandafter\xdef\csname glo@\@glo@label @flagfalse\endcsname{%
2400 \noexpand\global
2401 \noexpand\let\expandafter\noexpand
2402 \csname ifglo@\@glo@label @flag\endcsname\noexpand\iffalse
2403 }%
2404 \expandafter\xdef\csname glo@\@glo@label @flagtrue\endcsname{%
2405 \noexpand\global
2406 \noexpand\let\expandafter\noexpand
2407 \csname ifglo@\@glo@label @flag\endcsname\noexpand\iftrue
2408 }%
2409 \csname glo@\@glo@label @flagfalse\endcsname
```

Sort out any cross-referencing if required.

```
2410 \ifdefvoid\@glo@see
2411 {}%
2412 {%
2413 \protected@edef\@do@glsee{%
2414 \noexpand\@gls@fixbraces\noexpand\@glo@list\@glo@see
2415 \noexpand\@nil
2416 \noexpand\expandafter\noexpand\@glsee\noexpand\@glo@list{\@glo@label}}%
2417 \@do@glsee
2418 }%
```

Determine and store main part of the entry's index format.

```
2419 \ifignoredglossary\@glo@type
2420 {%
2421 \csdef{glo@\@glo@label @index}{}%
2422 }
2423 {%
2424 \do@glo@storeentry{\@glo@label}%
2425 }%
```

Define entry counters if enabled:

```
2426 \@newglossaryentry@defcounters
```

Add end hook in case another package wants to add extra keys.

```

2427 \newglossaryentryposthook
2428 }

aryentryprehook Allow extra information to be added to glossary entries:
2429 \newcommand*{\@newglossaryentryprehook}{}

ryentryposthook Allow extra information to be added to glossary entries:
2430 \newcommand*{\@newglossaryentryposthook}{}

try@defcounters
2431 \newcommand*{\@newglossaryentry@defcounters}{}

\glsmoveentry Moves entry whose label is given by first argument to the glossary named in the second argu-
ment.
2432 \newcommand*{\glsmoveentry}[2]{%
2433 \edef\@glo@thislabel{\glsdetoklabel{#1}}%
2434 \edef\@glo@type{\csname glo@\@glo@thislabel @type\endcsname}%
2435 \def\@glo@list{,%}
2436 \forglsentries[\@glo@type]{\@glo@label}%
2437 {%
2438 \ifdefequal\@glo@thislabel\@glo@label
2439 {}{\eappto\@glo@list{\@glo@label,%}}%
2440 }%
2441 \cslet\@glo@list\@glo@type{\@glo@list}%
2442 \csdef\@glo@\@glo@thislabel @type{#2}%
2443 }

ssaryentryfield Indicate what command should be used to display each entry in the glossary. (This enables
the glossaries-accsupp package to use \accsuppglossaryentryfield instead.)
2444 \ifglxindy
2445 \newcommand*{\@glossaryentryfield}{\string\glossentry}
2446 \else
2447 \newcommand*{\@glossaryentryfield}{\string\glossentry}
2448 \fi

rysubentryfield Indicate what command should be used to display each subentry in the glossary. (This en-
ables the glossaries-accsupp package to use \accsuppglossarysubentryfield instead.)
2449 \ifglxindy
2450 \newcommand*{\@glossarysubentryfield}{%
2451 \string\subglossentry}
2452 \else
2453 \newcommand*{\@glossarysubentryfield}{%
2454 \string\subglossentry}
2455 \fi

\@glo@storeentry \@glo@storeentry{<label>}

```

Determine the format to write the entry in the glossary output (.glo) file. The argument is the entry's label (should already have been de-tok'ed if required). The result is stored in \glo@<label>@index, where <label> is the entry's label. (This doesn't include any formatting or location information.)

```
2456 \newcommand{\@glo@storeentry}[1]{%
```

Escape makeindex/xindy special characters in the label:

```
2457 \edef\@glo@esclabel{#1}%
```

```
2458 \@gls@checkmkidxchars\@glo@esclabel
```

Get the sort string and escape any special characters

```
2459 \protected@edef\@glo@sort{\csname glo@#1@sort\endcsname}%
```

```
2460 \@gls@checkmkidxchars\@glo@sort
```

Same again for the name string. Escape any special characters in the prefix

```
2461 \@gls@checkmkidxchars\@glo@prefix
```

Get the parent, if one exists

```
2462 \edef\@glo@parent{\csname glo@#1@parent\endcsname}%
```

Write the information to the glossary file.

```
2463 \ifglxindy
```

Store using xindy syntax.

```
2464 \ifx\@glo@parent\@empty
```

Entry doesn't have a parent

```
2465 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
```

```
2466 (\string"\@glo@sort\string" %
```

```
2467 \string"\@glo@prefix\@glossaryentryfield{\@glo@esclabel}\string") %
```

```
2468 }%
```

```
2469 \else
```

Entry has a parent

```
2470 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
```

```
2471 \csname glo@\@glo@parent @index\endcsname
```

```
2472 (\string"\@glo@sort\string" %
```

```
2473 \string"\@glo@prefix\@glossarysubentryfield
```

```
2474 {\csname glo@#1@level\endcsname}{\@glo@esclabel}\string") %
```

```
2475 }%
```

```
2476 \fi
```

```
2477 \else
```

Store using makeindex syntax.

```
2478 \ifx\@glo@parent\@empty
```

Sanitize \@glo@prefix

```
2479 \@onelevel@sanitize\@glo@prefix
```

Entry doesn't have a parent

```
2480 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%
```

```
2481 \@glo@sort\@gls@actualchar\@glo@prefix
```

```
2482 \@glossaryentryfield{\@glo@esclabel}%
```

```

2483     }%
2484     \else
      Entry has a parent
2485     \expandafter\protected\xdef\csname glo@#1@index\endcsname{%
2486       \csname glo@\@glo@parent @index\endcsname\@gls@levelchar
2487       \@glo@sort\@gls@actualchar\@glo@prefix
2488       \@glossarysubentryfield
2489       {\csname glo@#1@level\endcsname}{\@glo@esclabel}%
2490     }%
2491     \fi
2492     \fi
2493 }

```

1.8 Resetting and unsetting entry flags

Each glossary entry is assigned a conditional of the form `\ifglo@<label>@flag` which determines whether or not the entry has been used (see also `\ifglsused` defined below). These flags can be set and unset using the following macros, but first we need to know if we're in `amsmath`'s align environment's measuring pass.

`@ifnotmeasuring`

```

2494 \AtBeginDocument{%
2495   \ifpackageloaded{amsmath}%
2496   {\let\gls@ifnotmeasuring\@gls@ifnotmeasuring}%
2497   }%
2498 }
2499 \newcommand*{\@gls@ifnotmeasuring}[1]{%
2500   \ifmeasuring@
2501   \else
2502     #1%
2503   \fi
2504 }
2505 \newcommand*\gls@ifnotmeasuring[1]{#1}

```

`\lspatchtabularx` Patch `\TX@trial` (as per David Carlisle's answer in <http://tex.stackexchange.com/a/94895>). This does nothing if `\TX@trial` hasn't been defined.

```

2506 \def\@gls@patchtabularx#1\hbox#2#3!!{%
2507   \def\TX@trial##1{#1\hbox{\let\glsunset\@gobble#2}#3}%
2508 }
2509 \newcommand*\glspatchtabularx{%
2510   \ifdef\TX@trial
2511   {%
2512     \expandafter\@gls@patchtabularx\TX@trial{##1}!!%
2513     \let\glspatchtabularx\relax
2514   }%
2515   }%
2516 }

```

`\glsreset` The command `\glsreset{<label>}` can be used to set the entry flag to indicate that it hasn't been used yet. The required argument is the entry label.

```
2517 \newcommand*{\glsreset}[1]{%
2518   \gls@ifnotmeasuring
2519   {%
2520     \glsdoifexists{#1}%
2521     {%
2522       \@glsreset{#1}%
2523     }%
2524   }%
2525 }
```

`\glslocalreset` As above, but with only a local effect:

```
2526 \newcommand*{\glslocalreset}[1]{%
2527   \gls@ifnotmeasuring
2528   {%
2529     \glsdoifexists{#1}%
2530     {%
2531       \@glslocalreset{#1}%
2532     }%
2533   }%
2534 }
```

`\glsunset` The command `\glsunset{<label>}` can be used to set the entry flag to indicate that it has been used. The required argument is the entry label.

```
2535 \newcommand*{\glsunset}[1]{%
2536   \gls@ifnotmeasuring
2537   {%
2538     \glsdoifexists{#1}%
2539     {%
2540       \@glsunset{#1}%
2541     }%
2542   }%
2543 }
```

`\glslocalunset` As above, but with only a local effect:

```
2544 \newcommand*{\glslocalunset}[1]{%
2545   \gls@ifnotmeasuring
2546   {%
2547     \glsdoifexists{#1}%
2548     {%
2549       \@glslocalunset{#1}%
2550     }%
2551   }%
2552 }
```

`\@glslocalunset` Local unset. This defaults to just `\@glslocalunset` but is changed by `\glsenableentrycount`.

```
2553 \newcommand*{\@glslocalunset}{\@glslocalunset}
```

`@@glslocalunset` Local unset without checks.

```

2554 \newcommand*{\@@glslocalunset}[1]{%
2555   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iftrue
2556 }

```

`\@glsunset` Global unset. This defaults to just `@@glsunset` but is changed by `\glsenableentrycount`.

```

2557 \newcommand*{\@glsunset}{\@@glsunset}

```

`\@@glsunset` Global unset without checks.

```

2558 \newcommand*{\@@glsunset}[1]{%
2559   \expandafter\global\csname glo@glsdetoklabel{#1}@flagtrue\endcsname
2560 }

```

`\@glslocalreset` Local reset. This defaults to just `@@glslocalreset` but is changed by `\glsenableentrycount`.

```

2561 \newcommand*{\@glslocalreset}{\@@glslocalreset}

```

`@@glslocalreset` Local reset without checks.

```

2562 \newcommand*{\@@glslocalreset}[1]{%
2563   \expandafter\let\csname ifglo@glsdetoklabel{#1}@flag\endcsname\iffalse
2564 }

```

`\@glsreset` Global reset. This defaults to just `@@glsreset` but is changed by `\glsenableentrycount`.

```

2565 \newcommand*{\@glsreset}{\@@glsreset}

```

`\@@glsreset` Global reset without checks.

```

2566 \newcommand*{\@@glsreset}[1]{%
2567   \expandafter\global\csname glo@glsdetoklabel{#1}@flagfalse\endcsname
2568 }

```

Reset all entries for the named glossaries (supplied in a comma-separated list). Syntax:

```

\glsresetall[<glossary-list>]

```

`\glsresetall`

```

2569 \newcommand*{\glsresetall}[1][\@glo@types]{%
2570   \forallglsentries[#1]{\@glsentry}%
2571   {%
2572     \glsreset{\@glsentry}%
2573   }%
2574 }

```

As above, but with only a local effect:

```

\glslocalresetall

```

```

2575 \newcommand*{\glslocalresetall}[1][\@glo@types]{%
2576   \forallglsentries[#1]{\@glsentry}%
2577   {%
2578     \glslocalreset{\@glsentry}%
2579   }%
2580 }

```

Unset all entries for the named glossaries (supplied in a comma-separated list). Syntax:
`\glsunsetall[⟨glossary-list⟩]`

`\glsunsetall`

```
2581 \newcommand*{\glsunsetall}[1][\@glo@types]{%
2582   \forallglsentries[#1]{\@glsentry}%
2583   {%
2584     \glsunset{\@glsentry}%
2585   }%
2586 }
```

As above, but with only a local effect:

`\glslocalunsetall`

```
2587 \newcommand*{\glslocalunsetall}[1][\@glo@types]{%
2588   \forallglsentries[#1]{\@glsentry}%
2589   {%
2590     \glslocalunset{\@glsentry}%
2591   }%
2592 }
```

1.9 Keeping Track of How Many Times an Entry Has Been Unset

Version 4.14 introduced `\glsenableentrycount` that keeps track of how many times an entry is marked as used. The counter is reset back to zero when the first use flag is reset. Note that although the word “counter” is used here, it’s not an actual \TeX counter or even an explicit \TeX count register but is just a macro. Any of the commands that use `\glsunset` or `\glslocalunset`, such as `\gls`, will automatically increment this value. Commands that don’t modify the first use flag (such as `\glstext` or `\glsentrytext`) don’t modify this value.

`\entry@defcounters` Define entry fields to keep track of how many times that entry has been marked as used.

```
2593 \newcommand*{\@newglossaryentry@defcounters}{%
2594   \csdef{glo@\@glo@label @currcount}{0}%
2595   \csdef{glo@\@glo@label @prevcount}{0}%
2596 }
```

`\enableentrycount` Enables tracking of how many times an entry has been marked as used.

```
2597 \newcommand*{\glsenableentrycount}{%
```

Enable new entry fields.

```
2598   \let\@newglossaryentry@defcounters\@newglossaryentry@defcounters
```

Disable `\newglossaryentry` in the document environment.

```
2599   \renewcommand*{\gls@defdocnewglossaryentry}{%
2600     \renewcommand*\newglossaryentry[2]{%
2601       \PackageError{glossaries}{\string\newglossaryentry\space
2602         may only be used in the preamble when entry counting has
```



```

2603     been activated}{If you use \string\glsenableentrycount\space
2604     you must place all entry definitions in the preamble not in
2605     the document environment}%
2606 }%
2607 }%

```

Define commands `\glsentrycurrcount` and `\glsentryprevcount` to access these new fields. Default to zero if undefined.

```

2608 \newcommand*{\glsentrycurrcount}[1]{%
2609   \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
2610   {0}{\@gls@entry@field{##1}{currcount}}%
2611 }%
2612 \newcommand*{\glsentryprevcount}[1]{%
2613   \ifcsundef{glo@\glsdetoklabel{##1}@prevcount}%
2614   {0}{\@gls@entry@field{##1}{prevcount}}%
2615 }%

```

Make the unset and reset functions also increment or reset the entry counter.

```

2616 \renewcommand*{\@glsunset}[1]{%
2617   \@@glsunset{##1}%
2618   \@gls@increment@currcount{##1}%
2619 }%
2620 \renewcommand*{\@glslocalunset}[1]{%
2621   \@@glslocalunset{##1}%
2622   \@gls@local@increment@currcount{##1}%
2623 }%
2624 \renewcommand*{\@glsreset}[1]{%
2625   \@@glsreset{##1}%
2626   \csgdef{glo@\glsdetoklabel{##1}@currcount}{0}%
2627 }%
2628 \renewcommand*{\@glslocalreset}[1]{%
2629   \@@glslocalreset{##1}%
2630   \csdef{glo@\glsdetoklabel{##1}@currcount}{0}%
2631 }%

```

Alter behaviour of `\cgl`s. (Only global unset is used if previous count was one as it doesn't make sense to have a local unset here given that the previous count was global.)

```

2632 \def\@cgl@s@##1##2[##3]{%
2633   \ifnum\glsentryprevcount{##2}=1\relax
2634     \cgl@sformat{##2}{##3}%
2635     \glsunset{##2}%
2636   \else
2637     \@gls@{##1}{##2}[##3]%
2638   \fi
2639 }%

```

Similarly for the analogous commands. No case change plural:

```

2640 \def\@cgl@spl@##1##2[##3]{%
2641   \ifnum\glsentryprevcount{##2}=1\relax
2642     \cgl@splformat{##2}{##3}%
2643     \glsunset{##2}%

```

```

2644 \else
2645     \@glspl@{##1}{##2}[##3]%
2646 \fi
2647 }%

```

First letter uppercase singular:

```

2648 \def\@cGls@##1##2[##3]{%
2649     \ifnum\glsentryprevcount{##2}=1\relax
2650         \cGlsformat{##2}{##3}%
2651         \glsunset{##2}%
2652     \else
2653         \@Gls@{##1}{##2}[##3]%
2654     \fi
2655 }%

```

First letter uppercase plural:

```

2656 \def\@cGlspl@##1##2[##3]{%
2657     \ifnum\glsentryprevcount{##2}=1\relax
2658         \cGlsplformat{##2}{##3}%
2659         \glsunset{##2}%
2660     \else
2661         \@Glspl@{##1}{##2}[##3]%
2662     \fi
2663 }%

```

Write information to aux file at the end of the document

```

2664 \AtEndDocument{\@gls@write@entrycounts}%

```

Fetch previous count information from aux file. (No check here to determine if the entry is still defined.)

```

2665 \renewcommand*{\@gls@entry@count}[2]{%
2666     \csgdef{glo@\glsdetoklabel{##1}@prevcount}{##2}%
2667 }%

```

\glsenableentrycount may only be used once and only in the preamble.

```

2668 \let\glsenableentrycount\relax
2669 }
2670 \@onlypreamble\glsenableentrycount

```

ement@currcount

```

2671 \newcommand*{\@gls@increment@currcount}[1]{%
2672     \csxdef{glo@\glsdetoklabel{##1}@currcount}{%
2673         \number\numexpr\glsentrycurrcount{##1}+1}%
2674 }

```

ement@currcount

```

2675 \newcommand*{\@gls@local@increment@currcount}[1]{%
2676     \csedef{glo@\glsdetoklabel{##1}@currcount}{%
2677         \number\numexpr\glsentrycurrcount{##1}+1}%
2678 }

```

ite@entrycounts Write the entry counts to the aux file. Use \immediate since this occurs right at the end of the document. Only write information for entries that have been used. (Some users have a file containing vast numbers of entries, many of which may not be used. There's no point writing information about the entries that haven't been used and it will only slow things down.)

```
2679 \newcommand*{\@gls@write@entrycounts}{%
2680   \immediate\write\@auxout
2681   {\string\providecommand*{\string\@gls@entry@count}[2]{}}%
2682   \forallglsentries{\@glsentry}{%
2683     \ifglsused{\@glsentry}%
2684     {\immediate\write\@auxout
2685       {\string\@gls@entry@count{\@glsentry}{\glsentrycurrcount{\@glsentry}}}%
2686     }%
2687   }%
2688 }
```

gls@entry@count Default behaviour is to ignore arguments. Activated by \glsenableentrycount.

```
2689 \newcommand*{\@gls@entry@count}[2]{}
```

\cgl Define command that works like \gls but behaves differently if the entry count function is enabled. (If not enabled, it behaves the same as \gls but issues a warning.)

```
2690 \newrobustcmd*{\cgl}{\@gls@hyp@opt\@cgl}
```

\@cgl Defined the un-starred form. Need to determine if there is a final optional argument

```
2691 \newcommand*{\@cgl}[2][ ]{%
2692   \new@ifnextchar[{\@cgl@{#1}{#2}}{\@cgl@{#1}{#2}[ ]}%
2693 }
```

\@cgl@ Read in the final optional argument. This defaults to same behaviour as \gls but issues a warning.

```
2694 \def\@cgl@#1#2[#3]{%
2695   \GlossariesWarning{\string\cgl\space is defaulting to
2696     \string\gls\space since you haven't enabled entry counting}%
2697   \@gls@{#1}{#2}[#3]%
2698 }
```

\cglformat Format used by \cgl if entry only used once on previous run. The first argument is the label, the second argument is the insert text.

```
2699 \newcommand*{\cglformat}[2]{%
2700   \ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}#2%
2701 }
```

\cGl Define command that works like \Gls but behaves differently if the entry count function is enabled. (If not enabled, it behaves the same as \Gls but issues a warning.)

```
2702 \newrobustcmd*{\cGl}{\@gls@hyp@opt\@cGl}
```

\@cGl Defined the un-starred form. Need to determine if there is a final optional argument

```
2703 \newcommand*{\@cGl}[2][ ]{%
```

```

2704 \new@ifnextchar[{\@cGls@{#1}{#2}}{\@cGls@{#1}{#2} []}%
2705 }

```

`\@cGls@` Read in the final optional argument. This defaults to same behaviour as `\Gls` but issues a warning.

```

2706 \def\@cGls@#1#2[#3]{%
2707 \GlossariesWarning{\string\cGls\space is defaulting to
2708 \string\Gls\space since you haven't enabled entry counting}%
2709 \@Gls@{#1}{#2}[#3]%
2710 }

```

`\cGlsformat` Format used by `\cGls` if entry only used once on previous run. The first argument is the label, the second argument is the insert text.

```

2711 \newcommand*{\cGlsformat}[2]{%
2712 \ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}#2%
2713 }

```

`\cglsp1` Define command that works like `\glsp1` but behaves differently if the entry count function is enabled. (If not enabled, it behaves the same as `\glsp1` but issues a warning.)

```

2714 \newrobustcmd*{\cglsp1}{\@gls@hyp@opt\@cglsp1}

```

`\@cglsp1` Defined the un-starred form. Need to determine if there is a final optional argument

```

2715 \newcommand*{\@cglsp1}[2] []{%
2716 \new@ifnextchar[{\@cglsp1@{#1}{#2}}{\@cglsp1@{#1}{#2} []}%
2717 }

```

`\@cglsp1@` Read in the final optional argument. This defaults to same behaviour as `\glsp1` but issues a warning.

```

2718 \def\@cglsp1@#1#2[#3]{%
2719 \GlossariesWarning{\string\cglsp1\space is defaulting to
2720 \string\glsp1\space since you haven't enabled entry counting}%
2721 \@glsp1@{#1}{#2}[#3]%
2722 }

```

`\cglsp1format` Format used by `\cglsp1` if entry only used once on previous run. The first argument is the label, the second argument is the insert text.

```

2723 \newcommand*{\cglsp1format}[2]{%
2724 \ifglshaslong{#1}{\glsp1entrylongpl{#1}}{\glsp1entryfirstplural{#1}}#2%
2725 }

```

`\cGlspl` Define command that works like `\Glspl` but behaves differently if the entry count function is enabled. (If not enabled, it behaves the same as `\Glspl` but issues a warning.)

```

2726 \newrobustcmd*{\cGlspl}{\@gls@hyp@opt\@cGlspl}

```

`\@cGlspl` Defined the un-starred form. Need to determine if there is a final optional argument

```

2727 \newcommand*{\@cGlspl}[2] []{%
2728 \new@ifnextchar[{\@cGlspl@{#1}{#2}}{\@cGlspl@{#1}{#2} []}%
2729 }

```

`\cGlspl@` Read in the final optional argument. This defaults to same behaviour as `\Glspl` but issues a warning.

```
2730 \def\cGlspl@#1#2[#3]{%
2731 \GlossariesWarning{\string\cGlspl\space is defaulting to
2732 \string\Glspl\space since you haven't enabled entry counting}%
2733 \cGlspl@{#1}{#2}[#3]%
2734 }
```

`\cGlsplformat` Format used by `\cGlspl` if entry only used once on previous run. The first argument is the label, the second argument is the insert text.

```
2735 \newcommand*{\cGlsplformat}[2]{%
2736 \ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}#2%
2737 }
```

1.10 Loading files containing glossary entries

Glossary entries can be defined in an external file. These external files can contain `\newglossaryentry` and `\newacronym` commands.¹

`\loadglsentries[⟨type⟩]{⟨filename⟩}`

This command will input the file using `\input`. The optional argument specifies to which glossary the entries should be assigned if they haven't used the type key. If the optional argument is not specified, the default glossary is used. Only those entries used in the document (via `\glslink`, `\gls`, `\glspl` and uppercase variants or `\glsadd` and `\glsaddall` will appear in the glossary). The mandatory argument is the filename (with or without `.tex` extension).

`\loadglsentries`

```
2738 \newcommand*{\loadglsentries}[2][\@gls@default]{%
2739 \let\@gls@default\glsdefaulttype
2740 \def\glsdefaulttype{#1}\input{#2}%
2741 \let\glsdefaulttype\@gls@default
2742 }
```

`\loadglsentries` can only be used in the preamble:

```
2743 \@onlypreamble{\loadglsentries}
```

1.11 Using glossary entries in the text

Any term that has been defined using `\newglossaryentry` (or `\newacronym`) can be displayed in the text (i.e. outside of the glossary) using one of the commands defined in this section. Unless you use `\glslink`, the way the term appears in the text is determined by `\glsdisplayfirst` (if it is the first time the term has been used) or `\glsdisplay` (for subsequent use). Any formatting commands (such as `\textbf` is governed by `\glstextformat`. By default this just displays the link text “as is”.

¹and any other valid \TeX code that can be used in the preamble.

`\glstextformat`

```
2744 \newcommand*{\glstextformat}[1]{#1}
```

`\glentryfmt` As from version 3.11a, the way in which an entry is displayed is now governed by `\glentryfmt`. This doesn't take any arguments. The required information is set by commands like `\gls`. To ensure backward compatibility, the default use the old `\glsdisplay` and `\glsdisplayfirst` style of commands

```
2745 \newcommand*{\glentryfmt}{%
2746   \@@gls@default@entryfmt\glsdisplayfirst\glsdisplay
2747 }
```

Format that provides backwards compatibility:

```
2748 \newcommand*{\@@gls@default@entryfmt}[2]{%
2749   \ifdefempty\glscustomtext
2750     {%
2751       \glsifplural
2752       {%
```

Plural form

```
2753       \glscapscase
2754       {%
```

Don't adjust case

```
2755       \ifglsused\glslabel
2756       {%
```

Subsequent use

```
2757       #2{\glentryplural{\glslabel}}%
2758       {\glentrydescplural{\glslabel}}%
2759       {\glentrysymbolplural{\glslabel}}{\glsinsert}%
2760     }%
2761     {%
```

First use

```
2762       #1{\glentryfirstplural{\glslabel}}%
2763       {\glentrydescplural{\glslabel}}%
2764       {\glentrysymbolplural{\glslabel}}{\glsinsert}%
2765     }%
2766     }%
2767     {%
```

Make first letter upper case

```
2768       \ifglsused\glslabel
2769       {%
```

Subsequent use. (Expansion was used in version 3.07 and below in case the name wasn't the first thing to be displayed, but now the user can sort out the upper casing in `\defglentryfmt`, which avoids the issues caused by fragile commands.)

```
2770       \ifbool{glscompatible-3.07}%
2771       {%
2772       \protected@edef\glo@etext{%
```

```

2773         #2{\glsentryplural{\glslabel}}%
2774         {\glsentrydescplural{\glslabel}}%
2775         {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2776     \xmakefirstuc\@glo@etext
2777 }%
2778 {%
2779     #2{\Glsentryplural{\glslabel}}%
2780     {\glsentrydescplural{\glslabel}}%
2781     {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2782 }%
2783 }%
2784 {%

```

First use

```

2785     \ifbool{glscompatible-3.07}%
2786     {%
2787         \protected@edef\@glo@etext{%
2788             #1{\glsentryfirstplural{\glslabel}}%
2789             {\glsentrydescplural{\glslabel}}%
2790             {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2791         \xmakefirstuc\@glo@etext
2792     }%
2793     {%
2794         #1{\Glsentryfirstplural{\glslabel}}%
2795         {\glsentrydescplural{\glslabel}}%
2796         {\glsentrysymbolplural{\glslabel}}{\glsinsert}%
2797     }%
2798 }%
2799 }%
2800 {%

```

Make all upper case

```

2801     \ifglsused\glslabel
2802     {%

```

Subsequent use

```

2803         \mfirstucMakeUppercase{#2{\glsentryplural{\glslabel}}%
2804         {\glsentrydescplural{\glslabel}}%
2805         {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2806     }%
2807     {%

```

First use

```

2808         \mfirstucMakeUppercase{#1{\glsentryfirstplural{\glslabel}}%
2809         {\glsentrydescplural{\glslabel}}%
2810         {\glsentrysymbolplural{\glslabel}}{\glsinsert}}%
2811     }%
2812 }%
2813 }%
2814 {%

```

Singular form

```
2815 \glscapscase
2816 {%
```

Don't adjust case

```
2817 \ifglused\glslabel
2818 {%
```

Subsequent use

```
2819 #2{\glentrytext{\glslabel}}%
2820 {\glentrydesc{\glslabel}}%
2821 {\glentrysymbol{\glslabel}}{\glinsert}%
2822 }%
2823 {%
```

First use

```
2824 #1{\glentryfirst{\glslabel}}%
2825 {\glentrydesc{\glslabel}}%
2826 {\glentrysymbol{\glslabel}}{\glinsert}%
2827 }%
2828 }%
2829 {%
```

Make first letter upper case

```
2830 \ifglused\glslabel
2831 {%
```

Subsequent use

```
2832 \ifbool{glcompatible-3.07}%
2833 {%
2834 \protected@edef\@glo@etext{%
2835 #2{\glentrytext{\glslabel}}%
2836 {\glentrydesc{\glslabel}}%
2837 {\glentrysymbol{\glslabel}}{\glinsert}}%
2838 \xmakefirstuc\@glo@etext
2839 }%
2840 {%
2841 #2{\Glsentrytext{\glslabel}}%
2842 {\glentrydesc{\glslabel}}%
2843 {\glentrysymbol{\glslabel}}{\glinsert}}%
2844 }%
2845 }%
2846 {%
```

First use

```
2847 \ifbool{glcompatible-3.07}%
2848 {%
2849 \protected@edef\@glo@etext{%
2850 #1{\glentryfirst{\glslabel}}%
2851 {\glentrydesc{\glslabel}}%
2852 {\glentrysymbol{\glslabel}}{\glinsert}}%
2853 \xmakefirstuc\@glo@etext
```



```

2854     }%
2855     {%
2856         #1{\Glsentryfirst{\glslabel}}%
2857         {\Glsentrydesc{\glslabel}}%
2858         {\Glsentrysymbol{\glslabel}}{\Glsinsert}%
2859     }%
2860 }%
2861 }%
2862 {%

```

Make all upper case

```

2863     \ifglsused\glslabel
2864     {%

```

Subsequent use

```

2865         \mfirstucMakeUppercase{#2{\Glsentrytext{\glslabel}}%
2866         {\Glsentrydesc{\glslabel}}%
2867         {\Glsentrysymbol{\glslabel}}{\Glsinsert}}%
2868     }%
2869     {%

```

First use

```

2870         \mfirstucMakeUppercase{#1{\Glsentryfirst{\glslabel}}%
2871         {\Glsentrydesc{\glslabel}}%
2872         {\Glsentrysymbol{\glslabel}}{\Glsinsert}}%
2873     }%
2874 }%
2875 }%
2876 }%
2877 {%

```

Custom text provided in \glsdisp

```

2878     \ifglsused{\glslabel}%
2879     {%

```

Subsequent use

```

2880         #2{\glscustomtext}%
2881         {\Glsentrydesc{\glslabel}}%
2882         {\Glsentrysymbol{\glslabel}}{}%
2883     }%
2884     {%

```

First use

```

2885         #1{\glscustomtext}%
2886         {\Glsentrydesc{\glslabel}}%
2887         {\Glsentrysymbol{\glslabel}}{}%
2888     }%
2889 }%
2890 }

```

`\glsentryfmt` Define a generic format that just uses the first, text, plural or first plural keys (or the custom text) with the insert text appended.

```

2891 \newcommand*{\glsgenentryfmt}{%
2892   \ifdefempty\glscustomtext
2893   {%
2894     \glssifplural
2895     {%

```

Plural form

```

2896     \glscapscase
2897     {%

```

Don't adjust case

```

2898     \ifglssused\glslabel
2899     {%

```

Subsequent use

```

2900     \glsentryplural{\glslabel}\glssinsert
2901     }%
2902     {%

```

First use

```

2903     \glsentryfirstplural{\glslabel}\glssinsert
2904     }%
2905     }%
2906     {%

```

Make first letter upper case

```

2907     \ifglssused\glslabel
2908     {%

```

Subsequent use.

```

2909     \Glsentryplural{\glslabel}\glssinsert
2910     }%
2911     {%

```

First use

```

2912     \Glsentryfirstplural{\glslabel}\glssinsert
2913     }%
2914     }%
2915     {%

```

Make all upper case

```

2916     \ifglssused\glslabel
2917     {%

```

Subsequent use

```

2918     \mfirstucMakeUppercase
2919     {\glsentryplural{\glslabel}\glssinsert}%
2920     }%
2921     {%

```

First use

```

2922     \mfirstucMakeUppercase
2923     {\glsentryfirstplural{\glslabel}\glssinsert}%

```

2924 }%

2925 }%

2926 }%

2927 {%

Singular form

2928 \glscapscase

2929 {%

Don't adjust case

2930 \ifglused\glslabel

2931 {%

Subsequent use

2932 \glentrytext{\glslabel}\glinsert

2933 }%

2934 {%

First use

2935 \glentryfirst{\glslabel}\glinsert

2936 }%

2937 }%

2938 {%

Make first letter upper case

2939 \ifglused\glslabel

2940 {%

Subsequent use

2941 \Glentrytext{\glslabel}\glinsert

2942 }%

2943 {%

First use

2944 \Glentryfirst{\glslabel}\glinsert

2945 }%

2946 }%

2947 {%

Make all upper case

2948 \ifglused\glslabel

2949 {%

Subsequent use

2950 \mfirstucMakeUppercase{\glentrytext{\glslabel}\glinsert}%

2951 }%

2952 {%

First use

2953 \mfirstucMakeUppercase{\glentryfirst{\glslabel}\glinsert}%

2954 }%

2955 }%

2956 }%

2957 }%

2958 {%

Custom text provided in \glsdisp. (The insert is most likely to be empty at this point.)

2959 \glscustomtext\glsinsert

2960 }%

2961 }

\glsngenacfmt Define a generic acronym format that uses the long and short keys (or their plurals) and \acrfullformat, \firstacronymfont and \acronymfont.

2962 \newcommand*{\glsngenacfmt}{%

2963 \ifdefempty\glscustomtext

2964 {%

2965 \ifglsused\glslabel

2966 {%

Subsequent use:

2967 \glsifplural

2968 {%

Subsequent plural form:

2969 \glscapscase

2970 {%

Subsequent plural form, don't adjust case:

2971 \acronymfont{\glsentryshortpl{\glslabel}}\glsinsert

2972 }%

2973 {%

Subsequent plural form, make first letter upper case:

2974 \acronymfont{\Glsentryshortpl{\glslabel}}\glsinsert

2975 }%

2976 {%

Subsequent plural form, all caps:

2977 \mfirstucMakeUppercase

2978 {\acronymfont{\glsentryshortpl{\glslabel}}\glsinsert}%

2979 }%

2980 }%

2981 {%

Subsequent singular form

2982 \glscapscase

2983 {%

Subsequent singular form, don't adjust case:

2984 \acronymfont{\glsentryshort{\glslabel}}\glsinsert

2985 }%

2986 {%

Subsequent singular form, make first letter upper case:

2987 \acronymfont{\Glsentryshort{\glslabel}}\glsinsert

2988 }%

2989 {%

Subsequent singular form, all caps:

```
2990      \mfirstucMakeUppercase
2991      {\acronymfont{\glentryshort{\glslabel}}\glsinsert}%
2992      }%
2993      }%
2994      }%
2995      {%
```

First use:

```
2996      \glsifplural
2997      {%
```

First use plural form:

```
2998      \glscapscase
2999      {%
```

First use plural form, don't adjust case:

```
3000      \genplacrfullformat{\glslabel}{\glsinsert}%
3001      }%
3002      {%
```

First use plural form, make first letter upper case:

```
3003      \Genplacrfullformat{\glslabel}{\glsinsert}%
3004      }%
3005      {%
```

First use plural form, all caps:

```
3006      \mfirstucMakeUppercase
3007      {\genplacrfullformat{\glslabel}{\glsinsert}}%
3008      }%
3009      }%
3010      {%
```

First use singular form

```
3011      \glscapscase
3012      {%
```

First use singular form, don't adjust case:

```
3013      \genacrfullformat{\glslabel}{\glsinsert}%
3014      }%
3015      {%
```

First use singular form, make first letter upper case:

```
3016      \Genacrfullformat{\glslabel}{\glsinsert}%
3017      }%
3018      {%
```

First use singular form, all caps:

```
3019      \mfirstucMakeUppercase
3020      {\genacrfullformat{\glslabel}{\glsinsert}}%
3021      }%
3022      }%
3023      }%
```

3024 }%

3025 {%

User supplied text.

3026 \glscustomtext

3027 }%

3028 }

genacrfullformat $\backslash\text{genacrfullformat}\{\langle label \rangle\}\{\langle insert \rangle\}$

The full format used by \glsgenacfmt (singular).

3029 \newcommand*{\genacrfullformat}[2]{%

3030 \glentrylong{#1}#2\space

3031 (\protect\firstacronymfont{\glentryshort{#1}})%

3032 }

Genacrfullformat $\backslash\text{Genacrfullformat}\{\langle label \rangle\}\{\langle insert \rangle\}$

As above but makes the first letter upper case.

3033 \newcommand*{\Genacrfullformat}[2]{%

3034 \protected@edef\gls@text{\genacrfullformat{#1}{#2}}%

3035 \xmakefirstuc\gls@text

3036 }

nplacrfullformat $\backslash\text{genplacrfullformat}\{\langle label \rangle\}\{\langle insert \rangle\}$

The full format used by \glsgenacfmt (plural).

3037 \newcommand*{\genplacrfullformat}[2]{%

3038 \glentrylongpl{#1}#2\space

3039 (\protect\firstacronymfont{\glentryshortpl{#1}})%

3040 }

nplacrfullformat $\backslash\text{Genplacrfullformat}\{\langle label \rangle\}\{\langle insert \rangle\}$

As above but makes the first letter upper case.

3041 \newcommand*{\Genplacrfullformat}[2]{%

3042 \protected@edef\gls@text{\genplacrfullformat{#1}{#2}}%

3043 \xmakefirstuc\gls@text

3044 }

glsdisplayfirst Deprecated. Kept for backward compatibility.

3045 \newcommand*{\glsdisplayfirst}[4]{#1#4}

`\glsdisplay` Deprecated. Kept for backward compatibility.

```
3046 \newcommand*{\glsdisplay}[4]{#1#4}
```

`\defglsdisplay` Deprecated. Kept for backward compatibility.

```
3047 \newcommand*{\defglsdisplay}[2][\glsdefaulttype]{%
3048   \GlossariesWarning{\string\defglsdisplay\space is now obsolete.^^J
3049   Use \string\defglsentryfmt\space instead}%
3050   \expandafter\def\csname gls@#1@display\endcsname##1##2##3##4{#2}%
3051   \edef\@gls@doentrydef{%
3052     \noexpand\defglsentryfmt[#1]{%
3053       \noexpand\ifcsdef{gls@#1@displayfirst}%
3054       {%
3055         \noexpand\@@gls@default@entryfmt
3056         {\noexpand\csuse{gls@#1@displayfirst}}%
3057         {\noexpand\csuse{gls@#1@display}}%
3058       }%
3059       {%
3060         \noexpand\@@gls@default@entryfmt
3061         {\noexpand\glsdisplayfirst}%
3062         {\noexpand\csuse{gls@#1@display}}%
3063       }%
3064     }%
3065   }%
3066   \@gls@doentrydef
3067 }
```

`\glsdisplayfirst` Deprecated. Kept for backward compatibility.

```
3068 \newcommand*{\defglsdisplayfirst}[2][\glsdefaulttype]{%
3069   \GlossariesWarning{\string\defglsdisplayfirst\space is now obsolete.^^J
3070   Use \string\defglsentryfmt\space instead}%
3071   \expandafter\def\csname gls@#1@displayfirst\endcsname##1##2##3##4{#2}%
3072   \edef\@gls@doentrydef{%
3073     \noexpand\defglsentryfmt[#1]{%
3074       \noexpand\ifcsdef{gls@#1@display}%
3075       {%
3076         \noexpand\@@gls@default@entryfmt
3077         {\noexpand\csuse{gls@#1@displayfirst}}%
3078         {\noexpand\csuse{gls@#1@display}}%
3079       }%
3080       {%
3081         \noexpand\@@gls@default@entryfmt
3082         {\noexpand\csuse{gls@#1@displayfirst}}%
3083         {\noexpand\glsdisplay}%
3084       }%
3085     }%
3086   }%
3087   \@gls@doentrydef
3088 }
```

Links to glossary entries

The links to glossary entries all have a first optional argument that can be used to change the format and counter of the associated entry number. Except for `\glslink` and `\glsdisp`, the commands like `\gls` have a final optional argument that can be used to insert additional text in the link (this will usually be appended, but can be redefined using `\defentryfmt`). It goes against the \TeX norm to have an optional argument after the mandatory arguments, but it makes more sense to write, say, `\gls{label}[‘s]` rather than, say, `\gls[append=‘s]{label}`. Since these control sequences are defined to include the final square bracket, spaces will be ignored after them. This is likely to lead to confusion as most users would not expect, say, `\gls{<label>}` to ignore following spaces, so `\new@ifnextchar` from the package is required.

The following keys can be used in the first optional argument. The counter key checks that the value is the name of a valid counter.

```
3089 \define@key{glslink}{counter}{%
3090   \ifcsundef{c@#1}%
3091   {%
3092     \PackageError{glossaries}%
3093     {There is no counter called ‘#1’}%
3094     {%
3095       The counter key should have the name of a valid counter
3096       as its value%
3097     }%
3098   }%
3099   {%
3100     \def\@gls@counter{#1}%
3101   }%
3102 }
```

The value of the format key should be the name of a command (without the initial backslash) that has a single mandatory argument which can be used to format the associated entry number.

```
3103 \define@key{glslink}{format}{%
3104   \def\@glsnumberformat{#1}}
```

The hyper key is a boolean key, it can either have the value true or false, and indicates whether or not to make a hyperlink to the relevant glossary entry. If hyper is false, an entry will still be made in the glossary, but the given text won't be a hyperlink.

```
3105 \define@boolkey{glslink}{hyper}[true]{}
```

Initialise hyper key.

```
3106 \ifdef{\hyperlink}{\KV@glslink@hypertrue}{\KV@glslink@hyperfalse}
```

The local key is a boolean key. If true this indicates that commands such as `\gls` should only do a local reset rather than a global one.

```
3107 \define@boolkey{glslink}{local}[true]{}
```

The original `\glsifhyper` command isn't particularly useful as it makes more sense to check the actual hyperlink setting rather than testing whether the starred or unstarred version has been used. Therefore, as from version 4.08, `\glsifhyper` is deprecated in favour of

`\glsifhyperon`. In case there is a particular need to know whether the starred or unstarred version was used, provide a new command that determines whether the *-version, +-version or unmodified version was used.

```
\glslinkvar{<unmodified case>}{<star case>}{<plus case>}
```

`\glslinkvar` Initialise to unmodified case.

```
3108 \newcommand*{\glslinkvar}[3]{#1}
```

`\glsifhyper` Now deprecated.

```
3109 \newcommand*{\glsifhyper}[2]{%
3110 \glslinkvar{#1}{#2}{#1}%
3111 \GlossariesWarning{\string\glsifhyper\space is deprecated. Did
3112 you mean \string\glsifhyperon\space or \string\glslinkvar?}%
3113 }
```

`\@gls@hyp@opt` Used by the commands such as `\glslink` to determine whether to modify the hyper option.

```
3114 \newcommand*{\@gls@hyp@opt}[1]{%
3115 \let\glslinkvar\@firstofthree
3116 \let\@gls@hyp@opt@cs#1\relax
3117 \@ifstar{\s@gls@hyp@opt}%
3118 {\@ifnextchar+{\@firstoftwo{\p@gls@hyp@opt}}{#1}}%
3119 }
```

`\s@gls@hyp@opt` Starred version

```
3120 \newcommand*{\s@gls@hyp@opt}[1] []{%
3121 \let\glslinkvar\@secondofthree
3122 \@gls@hyp@opt@cs[hyper=false,#1]}
```

`\p@gls@hyp@opt` Plus version

```
3123 \newcommand*{\p@gls@hyp@opt}[1] []{%
3124 \let\glslinkvar\@thirdofthree
3125 \@gls@hyp@opt@cs[hyper=true,#1]}
```

Syntax:

```
\glslink[<options>]{<label>}{<text>}
```

Display `<text>` in the document, and add the entry information for `<label>` into the relevant glossary. The optional argument should be a key value list using the `\glslink` keys defined above.

There is also a starred version:

```
\glslink*{<options>}{<label>}{<text>}
```

which is equivalent to `\glslink[hyper=false,<options>]{<label>}{<text>}`

First determine which version is being used:

`\glslink`

```
3126 \newrobustcmd*{\glslink}{%  
3127 \@gls@hyp@opt\@gls@@link  
3128 }
```

`\@gls@@link` The main part of the business is in `\@gls@link` which shouldn't check if the term is defined as it's called by `\gls` etc which also perform that check.

```
3129 \newcommand*{\@gls@@link}[3][\@gls@link]{%  
3130 \glsdoifexistsordo{#2}%  
3131 {%  
3132 \let\do@gls@link@checkfirsthyper\relax  
3133 \@gls@link[#1]{#2}{#3}%  
3134 }%
```

Display the specified text. (The entry doesn't exist so there's nothing to link it to.)

```
3135 \glstextformat{#3}%  
3136 }%
```

```
3137 \glspostlinkhook  
3138 }
```

`glspostlinkhook`

```
3139 \newcommand*{\glspostlinkhook}{}%
```

`checkfirsthyper` Check for first use and switch off hyper key if hyperlink not wanted. (Should be off if first use and `hyper=false` is on or if first use and both the entry is in an acronym list and the `acrfootnote` setting is on.) This assumes the glossary type is stored in `\glstype` and the label is stored in `\glslabel`.

```
3140 \newcommand*{\@gls@link@checkfirsthyper}{%  
3141 \ifglsused{\glslabel}%  
3142 {%  
3143 }%  
3144 {%  
3145 \gls@checkisacronymlist\glstype  
3146 \ifglshyperfirst  
3147 \if@glsisacronymlist  
3148 \ifglsacrfootnote  
3149 \KV@glslink@hyperfalse  
3150 \fi  
3151 \fi  
3152 \else  
3153 \KV@glslink@hyperfalse  
3154 \fi  
3155 }%
```

Allow user to hook into this

```
3156 \glslinkcheckfirsthyperhook  
3157 }
```

`checkfirsthyperhook` Allow used to hook into the `\@gls@link@checkfirsthyper` macro
3158 `\newcommand*\@glslinkcheckfirsthyperhook{}\}`

`linkpostsetkeys`
3159 `\newcommand*\@glslinkpostsetkeys{}\}`

`\glsifhyperon` Check the value of the hyper key:
3160 `\newcommand{\glsifhyperon}[2]{\ifKV@glslink@hyper#1\else#2\fi}`

`disablehyperinlist` Disable hyperlink if in the “nohyper” list.
3161 `\newcommand*\@do@glsdisablehyperinlist{}\%`
3162 `\expandafter\DTLifinlist\expandafter{\gls@type}\@gls@nohyperlist}%`
3163 `{\KV@glslink@hyperfalse}\}`
3164 `}`

`\let@glslink@opts` Hook to set default options for `\@glslink`.
3165 `\newcommand*\@gls@setdefault@glslink@opts{}\}`

`\@gls@link`
3166 `\def\@gls@link[#1]#2#3{\%`
Inserting `\leavevmode` suggested by Donald Arseneau (avoids problem with tabularx).
3167 `\leavevmode`
3168 `\edef\glslabel{\glsdetoklabel{#2}}%`
Save options in `\@gls@link@opts` and label in `\@gls@link@label`
3169 `\def\@gls@link@opts{#1}%`
3170 `\let\@gls@link@label\glslabel`
3171 `\def\@glsnumberformat{glsnumberformat}%`
3172 `\edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%`
If this is in one of the “nohypertypes” glossaries, suppress the hyperlink by default
3173 `\edef\gls@type{\csname glo@\glslabel @type\endcsname}%`
Save original setting
3174 `\let\org@ifKV@glslink@hyper\ifKV@glslink@hyper`
Set defaults:
3175 `\@gls@setdefault@glslink@opts`
Switch off hyper setting if the glossary type has been identified in nohyperlist.
3176 `\@do@glsdisablehyperinlist`
Macros must set this before calling `\@gls@link`. The commands that check the first use flag should set this to `\@gls@link@checkfirsthyper` otherwise it should be set to `\relax`.
3177 `\do@gls@link@checkfirsthyper`
3178 `\setkeys{glslink}{#1}%`
Add a hook for the user to customise things after the keys have been set.
3179 `\glslinkpostsetkeys`

```

Store the entry's counter in \theglsentrycounter
3180 \gls@saveentrycounter

Define sort key if necessary:
3181 \gls@setsort{\glslabel}%

(De-tok'ing done by \@do@wrglossary)
3182 \@do@wrglossary{#2}%
3183 \ifKV@glslink@hyper
3184 \glslink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
3185 \else

3186 \glsdonohyperlink{\glolinkprefix\glslabel}{\glstextformat{#3}}%
3187 \fi

Restore original setting
3188 \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
3189 }

\glolinkprefix
3190 \newcommand*{\glolinkprefix}{glo:}

glsentrycounter Set default value of entry counter
3191 \def\glsentrycounter{\glscounter}%

saveentrycounter Need to check if using equation counter in align environment:
3192 \newcommand*{\gls@saveentrycounter}{%
3193 \def\gls@Hcounter{}%

Are we using equation counter?
3194 \ifthenelse{\equal{\gls@counter}{equation}}{%
3195 {

If we're in align environment, \xatlevel@ will be defined. (Can't test for \@currentenv as
may be inside an inner environment.)
3196 \ifcsundef{xatlevel@}%
3197 {%
3198 \edef\theglsentrycounter{\expandafter\noexpand
3199 \csname the\gls@counter\endcsname}%
3200 }%
3201 {%
3202 \ifx\xatlevel@\@empty
3203 \edef\theglsentrycounter{\expandafter\noexpand
3204 \csname the\gls@counter\endcsname}%
3205 \else
3206 \savecounters@
3207 \advance\c@equation by 1\relax
3208 \edef\theglsentrycounter{\csname the\gls@counter\endcsname}%

```

Check if hyperref version of this counter

```

3209     \ifcsundef{theH\@gls@counter}%
3210     {%
3211         \def\@gls@Hcounter{\theglsentrycounter}%
3212     }%
3213     {%
3214         \def\@gls@Hcounter{\csname theH\@gls@counter\endcsname}%
3215     }%
3216     \protected@edef\theHglentrycounter{\@gls@Hcounter}%
3217     \restorecounters@
3218 \fi
3219 }%
3220 }%
3221 {%

```

Not using equation counter so no special measures:

```

3222     \edef\theglsentrycounter{\expandafter\noexpand
3223         \csname the\@gls@counter\endcsname}%
3224 }%

```

Check if hyperref version of this counter

```

3225 \ifx\@gls@Hcounter\@empty
3226 \ifcsundef{theH\@gls@counter}%
3227 {%
3228     \def\theHglentrycounter{\theglsentrycounter}%
3229 }%
3230 {%
3231     \protected@edef\theHglentrycounter{\expandafter\noexpand
3232         \csname theH\@gls@counter\endcsname}%
3233 }%
3234 \fi
3235 }

```

`t@glo@numformat` Set the formatting information in the format required by `makeindex`. The first argument is the format specified by the user (via the `format` key), the second argument is the name of the counter used to indicate the location, the third argument is a control sequence which stores the required format and the fourth argument (new to v3.0) is the hyper-prefix.

```

3236 \def\@set@glo@numformat#1#2#3#4{%
3237     \expandafter\@glo@check@mkidxrangechar#3\@nil
3238     \protected@edef#1{%
3239         \@glo@prefix setentrycounter[#4]{#2}%
3240         \expandafter\string\csname\@glo@suffix\endcsname
3241     }%
3242     \@gls@checkmkidxchars#1%
3243 }

```

Check to see if the given string starts with a (or). If it does set `\@glo@prefix` to the starting character, and `\@glo@suffix` to the rest (or `glsnumberformat` if there is nothing else), otherwise set `\@glo@prefix` to nothing and `\@glo@suffix` to all of it.

```

3244 \def\@glo@check@mkidxrangechar#1#2\@nil{%
3245 \if#1(\relax
3246   \def\@glo@prefix{()%
3247   \if\relax#2\relax
3248     \def\@glo@suffix{glsnumberformat}%
3249   \else
3250     \def\@glo@suffix{#2}%
3251   \fi
3252 \else
3253   \if#1)\relax
3254     \def\@glo@prefix{}}}%
3255   \if\relax#2\relax
3256     \def\@glo@suffix{glsnumberformat}%
3257   \else
3258     \def\@glo@suffix{#2}%
3259   \fi
3260 \else
3261   \def\@glo@prefix{}\def\@glo@suffix{#1#2}%
3262 \fi
3263 \fi}

```

`\@gls@escbsdq` Escape backslashes and double quote marks. The argument must be a control sequence.

```

3264 \newcommand*{\@gls@escbsdq}[1]{%
3265   \def\@gls@checkedmkidx{%
3266     \let\gls@xdystring=#1\relax
3267     \@onelevel@sanitize\gls@xdystring
3268     \edef\do@gls@xdycheckbackslash{%
3269       \noexpand\@gls@xdycheckbackslash\gls@xdystring\noexpand\@nil
3270       \@backslashchar\@backslashchar\noexpand\null}%
3271     \do@gls@xdycheckbackslash
3272     \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%
3273     \def\@gls@checkedmkidx{%
3274       \expandafter\@gls@xdycheckquote\gls@xdystring\@nil""\null
3275       \expandafter\@gls@updatechecked\@gls@checkedmkidx{\gls@xdystring}%

```

Unsanitize\gls@numberpage,\gls@alphpage,\gls@Alphpage and\gls@romanpage (thanks to David Carlisle for the suggestion.)

```

3276 \@for\@gls@tmp:=\gls@protected@pagefmts\do
3277 {%
3278   \edef\@gls@sanitized@tmp{\expandafter\@gobble\string\\expandonce\@gls@tmp}%
3279   \@onelevel@sanitize\@gls@sanitized@tmp
3280   \edef\gls@dostsubst{%
3281     \noexpand\DTLsubstituteall\noexpand\gls@xdystring
3282     {\@gls@sanitized@tmp}{\expandonce\@gls@tmp}%
3283   }%
3284   \gls@dostsubst
3285 }%

```

Assign to required control sequence

```

3286 \let#1=\gls@xdystring

```

3287 }

Catch special characters (argument must be a control sequence):

checkmkidxchars

```
3288 \newcommand{\@gls@checkmkidxchars}[1]{%
3289   \ifglxsindy
3290     \@gls@escbsdq{#1}%
3291   \else
3292     \def\@gls@checkedmkidx{%
3293       \expandafter\@gls@checkquote#1\@nil""\null
3294       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3295     \def\@gls@checkedmkidx{%
3296       \expandafter\@gls@checkescquote#1\@nil\\"\null
3297       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3298     \def\@gls@checkedmkidx{%
3299       \expandafter\@gls@checkescactual#1\@nil??\null
3300       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3301     \def\@gls@checkedmkidx{%
3302       \expandafter\@gls@checkactual#1\@nil??\null
3303       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3304     \def\@gls@checkedmkidx{%
3305       \expandafter\@gls@checkbar#1\@nil||\null
3306       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3307     \def\@gls@checkedmkidx{%
3308       \expandafter\@gls@checkescbar#1\@nil\\|\null
3309       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3310     \def\@gls@checkedmkidx{%
3311       \expandafter\@gls@checklevel#1\@nil!!\null
3312       \expandafter\@gls@updatechecked\@gls@checkedmkidx{#1}%
3313   \fi
3314 }
```

Update the control sequence and strip trailing \@nil:

s@updatechecked

```
3315 \def\@gls@updatechecked#1\@nil#2{\def#2{#1}}
```

\@gls@tmpb Define temporary token

```
3316 \newtoks\@gls@tmpb
```

@gls@checkquote Replace " with "" since " is a makeindex special character.

```
3317 \def\@gls@checkquote#1"#2"#3\null{%
3318   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3319   \toks@={#1}%
3320   \ifx\@nil#2\null
3321   \ifx\@nil#3\null
3322     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3323   \def\@gls@checkquote{\relax}%
3324   \else
```

```

3325 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3326 \@gls@quotechar\@gls@quotechar\@gls@quotechar\@gls@quotechar}%
3327 \def\@@gls@checkquote{\@gls@checkquote#3\null}%
3328 \fi
3329 \else
3330 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3331 \@gls@quotechar\@gls@quotechar}%
3332 \ifx\null#3\null
3333 \def\@@gls@checkquote{\@gls@checkquote#2""\null}%
3334 \else
3335 \def\@@gls@checkquote{\@gls@checkquote#2"#3\null}%
3336 \fi
3337 \fi
3338 \@@gls@checkquote
3339 }

```

s@checkescquote Do the same for \":

```

3340 \def\@gls@checkescquote#1\"#2\"#3\null{%
3341 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3342 \toks@={#1}%
3343 \ifx\null#2\null
3344 \ifx\null#3\null
3345 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3346 \def\@@gls@checkescquote{\relax}%
3347 \else
3348 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3349 \@gls@quotechar\string\"@gls@quotechar
3350 \@gls@quotechar\string\"@gls@quotechar}%
3351 \def\@@gls@checkescquote{\@gls@checkescquote#3\null}%
3352 \fi
3353 \else
3354 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3355 \@gls@quotechar\string\"@gls@quotechar}%
3356 \ifx\null#3\null
3357 \def\@@gls@checkescquote{\@gls@checkescquote#2\"\" \null}%
3358 \else
3359 \def\@@gls@checkescquote{\@gls@checkescquote#2\"#3\null}%
3360 \fi
3361 \fi
3362 \@@gls@checkescquote
3363 }

```

@checkescactual Similarly for \? (which is replaces @ as makeindex's special character):

```

3364 \def\@gls@checkescactual#1\?#2\?#3\null{%
3365 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3366 \toks@={#1}%
3367 \ifx\null#2\null
3368 \ifx\null#3\null
3369 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%

```



```

3370 \def\@gls@checkescactual{\relax}%
3371 \else
3372 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3373 \@gls@quotechar\string"\@gls@actualchar
3374 \@gls@quotechar\string"\@gls@actualchar}%
3375 \def\@gls@checkescactual{\@gls@checkescactual#3\null}%
3376 \fi
3377 \else
3378 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3379 \@gls@quotechar\string"\@gls@actualchar}%
3380 \ifx\null#3\null
3381 \def\@gls@checkescactual{\@gls@checkescactual#2\?\?\null}%
3382 \else
3383 \def\@gls@checkescactual{\@gls@checkescactual#2\?#3\null}%
3384 \fi
3385 \fi
3386 \@gls@checkescactual
3387 }

```

`gls@checkescbar` Similarly for `\|`:

```

3388 \def\@gls@checkescbar#1\|#2\|#3\null{%
3389 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3390 \toks@={#1}%
3391 \ifx\null#2\null
3392 \ifx\null#3\null
3393 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3394 \def\@gls@checkescbar{\relax}%
3395 \else
3396 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3397 \@gls@quotechar\string"\@gls@encapchar
3398 \@gls@quotechar\string"\@gls@encapchar}%
3399 \def\@gls@checkescbar{\@gls@checkescbar#3\null}%
3400 \fi
3401 \else
3402 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3403 \@gls@quotechar\string"\@gls@encapchar}%
3404 \ifx\null#3\null
3405 \def\@gls@checkescbar{\@gls@checkescbar#2\|\|\null}%
3406 \else
3407 \def\@gls@checkescbar{\@gls@checkescbar#2\|#3\null}%
3408 \fi
3409 \fi
3410 \@gls@checkescbar
3411 }

```

`s@checkesclevel` Similarly for `\!`:

```

3412 \def\@gls@checkesclevel#1\!#2\!#3\null{%
3413 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3414 \toks@={#1}%

```

```

3415 \ifx\null#2\null
3416 \ifx\null#3\null
3417 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3418 \def\@gls@checkesclevel{\relax}%
3419 \else
3420 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3421 \gls@quotechar\string"\@gls@levelchar
3422 \gls@quotechar\string"\@gls@levelchar}%
3423 \def\@gls@checkesclevel{\@gls@checkesclevel#3\null}%
3424 \fi
3425 \else
3426 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3427 \gls@quotechar\string"\@gls@levelchar}%
3428 \ifx\null#3\null
3429 \def\@gls@checkesclevel{\@gls@checkesclevel#2!!\null}%
3430 \else
3431 \def\@gls@checkesclevel{\@gls@checkesclevel#2!#3\null}%
3432 \fi
3433 \fi
3434 \@gls@checkesclevel
3435 }

```

\@gls@checkbar and for |:

```

3436 \def\@gls@checkbar#1|#2|#3\null{%
3437 \gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3438 \toks@={#1}%
3439 \ifx\null#2\null
3440 \ifx\null#3\null
3441 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3442 \def\@gls@checkbar{\relax}%
3443 \else
3444 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3445 \gls@quotechar\gls@encapchar\gls@quotechar\gls@encapchar}%
3446 \def\@gls@checkbar{\@gls@checkbar#3\null}%
3447 \fi
3448 \else
3449 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3450 \gls@quotechar\gls@encapchar}%
3451 \ifx\null#3\null
3452 \def\@gls@checkbar{\@gls@checkbar#2||\null}%
3453 \else
3454 \def\@gls@checkbar{\@gls@checkbar#2|#3\null}%
3455 \fi
3456 \fi
3457 \@gls@checkbar
3458 }

```

@gls@checklevel and for !:

```

3459 \def\@gls@checklevel#1!#2!#3\null{%

```

```

3460 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3461 \toks@={#1}%
3462 \ifx\null#2\null
3463   \ifx\null#3\null
3464     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3465     \def\@gls@checklevel{\relax}%
3466   \else
3467     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3468       \@gls@quotechar\@gls@levelchar\@gls@quotechar\@gls@levelchar}%
3469     \def\@gls@checklevel{\@gls@checklevel#3\null}%
3470   \fi
3471 \else
3472   \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3473     \@gls@quotechar\@gls@levelchar}%
3474   \ifx\null#3\null
3475     \def\@gls@checklevel{\@gls@checklevel#2!!\null}%
3476   \else
3477     \def\@gls@checklevel{\@gls@checklevel#2!#3\null}%
3478   \fi
3479 \fi
3480 \@gls@checklevel
3481 }

```

gls@checkactual and for ?:

```

3482 \def\@gls@checkactual#1?#2?#3\null{%
3483   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3484   \toks@={#1}%
3485   \ifx\null#2\null
3486     \ifx\null#3\null
3487       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3488       \def\@gls@checkactual{\relax}%
3489     \else
3490       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3491         \@gls@quotechar\@gls@actualchar\@gls@quotechar\@gls@actualchar}%
3492       \def\@gls@checkactual{\@gls@checkactual#3\null}%
3493     \fi
3494   \else
3495     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3496       \@gls@quotechar\@gls@actualchar}%
3497     \ifx\null#3\null
3498       \def\@gls@checkactual{\@gls@checkactual#2??\null}%
3499     \else
3500       \def\@gls@checkactual{\@gls@checkactual#2?#3\null}%
3501     \fi
3502   \fi
3503   \@gls@checkactual
3504 }

```

s@xdycheckquote As before but for use with xindy

```

3505 \def\@gls@xdycheckquote#1"#2"#3\null{%
3506   \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
3507   \toks@={#1}%
3508   \ifx\null#2\null
3509     \ifx\null#3\null
3510       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
3511       \def\@gls@xdycheckquote{\relax}%
3512     \else
3513       \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3514         \string"\string"}%
3515       \def\@gls@xdycheckquote{\@gls@xdycheckquote#3\null}%
3516     \fi
3517   \else
3518     \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
3519       \string"}%
3520     \ifx\null#3\null
3521       \def\@gls@xdycheckquote{\@gls@xdycheckquote#2""\null}%
3522     \else
3523       \def\@gls@xdycheckquote{\@gls@xdycheckquote#2"#3\null}%
3524     \fi
3525   \fi
3526   \@gls@xdycheckquote
3527 }

```

ycheckbackslash Need to escape all backslashes for xindy. Define command that will define \@gls@xdycheckbackslash

```

3528 \edef\def\@gls@xdycheckbackslash{%
3529   \noexpand\def\noexpand\@gls@xdycheckbackslash##1\@backslashchar
3530     ##2\@backslashchar##3\noexpand\null{%
3531     \noexpand\@gls@tmpb=\noexpand\expandafter
3532       {\noexpand\@gls@checkedmkidx}%
3533     \noexpand\toks@={##1}%
3534     \noexpand\ifx\noexpand\null##2\noexpand\null
3535       \noexpand\ifx\noexpand\null##3\noexpand\null
3536         \noexpand\edef\noexpand\@gls@checkedmkidx{%
3537           \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
3538         \noexpand\def\noexpand\@gls@xdycheckbackslash{\relax}%
3539       \noexpand\else
3540         \noexpand\edef\noexpand\@gls@checkedmkidx{%
3541           \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3542           \@backslashchar\@backslashchar\@backslashchar\@backslashchar}%
3543       \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3544         \noexpand\@gls@xdycheckbackslash##3\noexpand\null}%
3545       \noexpand\fi
3546     \noexpand\else
3547       \noexpand\edef\noexpand\@gls@checkedmkidx{%
3548         \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
3549         \@backslashchar\@backslashchar}%
3550     \noexpand\ifx\noexpand\null##3\noexpand\null
3551       \noexpand\def\noexpand\@gls@xdycheckbackslash{%

```

```

3552      \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3553      \@backslashchar\noexpand\null}%
3554 \noexpand\else
3555      \noexpand\def\noexpand\@gls@xdycheckbackslash{%
3556      \noexpand\@gls@xdycheckbackslash##2\@backslashchar
3557      ##3\noexpand\null}%
3558 \noexpand\fi
3559 \noexpand\fi
3560 \noexpand\@gls@xdycheckbackslash
3561 }%
3562 }

```

Now go ahead and define \@gls@xdycheckbackslash

```

3563 \def@gls@xdycheckbackslash

```

lgsdohypertarget

```

3564 \newlength\gls@tmplen
3565 \newcommand*\lgsdohypertarget}[2]{%
3566   \settoheight{\gls@tmplen}{#2}%
3567   \raisebox{\gls@tmplen}{\hypertarget{#1}{}}#2%
3568 }

```

\lgsdohyperlink

```

3569 \newcommand*\lgsdohyperlink}[2]{%
3570   \hyperlink{#1}{#2}%
3571 }

```

lgsdonohyperlink

```

3572 \newcommand*\lgsdonohyperlink}[2]{#2}

```

\@glslink If \hyperlink is not defined \@glslink ignores its first argument and just does the second argument, otherwise it is equivalent to \hyperlink.

```

3573 \ifcsundef{hyperlink}%
3574 {%
3575   \let\@glslink\lgsdonohyperlink
3576 }%
3577 {%
3578   \let\@glslink\lgsdohyperlink
3579 }

```

\@glstarget If \hypertarget is not defined, \@glstarget ignores its first argument and just does the second argument, otherwise it is equivalent to \hypertarget.

```

3580 \ifcsundef{hypertarget}%
3581 {%
3582   \let\@glstarget\@secondoftwo
3583 }%
3584 {%
3585   \let\@glstarget\lgsdohypertarget
3586 }

```

Glossary hyperlinks can be disabled using `\glsdisablehyper` (effect can be localised):

`\glsdisablehyper`

```
3587 \newcommand{\glsdisablehyper}{%
3588   \KV@glslink@hyperfalse
3589   \let\@glslink\glsdonohyperlink
3590   \let\@glstarget\@secondoftwo
3591 }
```

Glossary hyperlinks can be enabled using `\glsenablehyper` (effect can be localised):

`\glsenablehyper`

```
3592 \newcommand{\glsenablehyper}{%
3593   \KV@glslink@hypertrue
3594   \let\@glslink\glsdohyperlink
3595   \let\@glstarget\glsdohypertarget
3596 }
```

Provide some convenience commands if not already defined:

```
3597 \providecommand{\@firstofthree}[3]{#1}
3598 \providecommand{\@secondofthree}[3]{#2}
```

Syntax:

`\gls[<options>]{<label>}[<insert text>]`

Link to glossary entry using singular form. The link text is taken from the value of the text or first keys used when the entry was defined.

The first optional argument is a key-value list, the same as `\glslink`, the mandatory argument is the entry label. After the mandatory argument, there is another optional argument to insert extra text in the link text (the location of the inserted text is governed by `\glsdisplay` and `\glsdisplayfirst`). As with `\glslink` there is a starred version which is the same as the unstarred version but with the hyper key set to `false`. (Additional options can also be specified in the first optional argument.)

First determine which version is being used:

`\gls`

```
3599 \newrobustcmd*{\gls}{\@gls@hyp@opt\@gls}
```

Defined the un-starred form. Need to determine if there is a final optional argument

`\@gls`

```
3600 \newcommand*{\@gls}[2][ ]{%
3601   \new@ifnextchar[{\@gls@{#1}{#2}}{\@gls@{#1}{#2}[ ]}%
3602 }
```

`\@gls@` Read in the final optional argument:

```
3603 \def\@gls@#1#2[#3]{%
3604   \glsdoifexists{#2}%
3605   {%
3606     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
```

```

3607 \let\glsifplural\@secondoftwo
3608 \let\glscapscase\@firstofthree
3609 \let\glscustomtext\@empty
3610 \def\glsinsert{#3}%

```

Determine what the link text should be (this is stored in \@glo@text) Note that \@gls@link sets \glstype.

```

3611 \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%

```

Call \@gls@link. If footnote package option has been used and the glossary type is \acronymtype, suppress hyperlink for first use. Likewise if the hyperfirst=false package option is used.

```

3612 \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3613 \ifKV@glslink@local
3614 \glslocalunset{#2}%
3615 \else
3616 \glsunset{#2}%
3617 \fi
3618 }%

```

```

3619 \glspostlinkhook
3620 }

```

\Gls behaves like \gls, but the first letter of the link text is converted to uppercase (note that if the first letter has an accent, the accented letter will need to be grouped when you define the entry). It is mainly intended for terms that start a sentence:

\Gls

```

3621 \newrobustcmd*{\Gls}{\@gls@hyp@opt\@Gls}

```

Defined the un-starred form. Need to determine if there is a final optional argument

```

3622 \newcommand*{\@Gls}[2][\@Gls@{#1}{#2}]{\@Gls@{#1}{#2}[]}
3623 \new@ifnextchar[\@Gls@{#1}{#2}]{\@Gls@{#1}{#2}[]}
3624 }

```

\@Gls@ Read in the final optional argument:

```

3625 \def\@Gls@#1#2[#3]{%
3626 \glsdoifexists{#2}%
3627 {%
3628 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3629 \let\glsifplural\@secondoftwo
3630 \let\glscapscase\@secondofthree
3631 \let\glscustomtext\@empty
3632 \def\glsinsert{#3}%

```

Determine what the link text should be (this is stored in \@glo@text) Note that \@gls@link sets \glstype.

```

3633 \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%

```

Call `\@gls@link` If footnote package option has been used and the glossary type is `\acronymtype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```
3634 \gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3635 \ifKV@glslink@local
3636 \glslocalunset{#2}%
3637 \else
3638 \glsunset{#2}%
3639 \fi
3640 }%
```

```
3641 \glspostlinkhook
3642 }
```

`\GLS` behaves like `\gls`, but the link text is converted to uppercase:

`\GLS`

```
3643 \newrobustcmd*{\GLS}{\@gls@hyp@opt\@GLS}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3644 \newcommand*{\@GLS}[2][\@GLS@{#1}{#2}]{\@GLS@{#1}{#2}[\@GLS@{#1}{#2}[]]}%
3645 \new@ifnextchar[\@GLS@{#1}{#2}]{\@GLS@{#1}{#2}[\@GLS@{#1}{#2}[]]}%
3646 }
```

`\@GLS@` Read in the final optional argument:

```
3647 \def\@GLS@#1#2[#3]{%
3648 \glsdoifexists{#2}%
3649 {%
3650 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3651 \let\glsifplural\@secondoftwo
3652 \let\glscapscase\@thirdofthree
3653 \let\glscustomtext\@empty
3654 \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in `\@glo@text`). Note that `\@gls@link` sets `\glstype`.

```
3655 \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
```

Call `\@gls@link` If footnote package option has been used and the glossary type is `\acronymtype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```
3656 \gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3657 \ifKV@glslink@local
3658 \glslocalunset{#2}%
3659 \else
3660 \glsunset{#2}%
3661 \fi
3662 }%
```



```

3663 \glspostlinkhook
3664 }

```

`\glspl` behaves in the same way as `\gls` except it uses the plural form.

`\glspl`

```

3665 \newrobustcmd*{\glspl}{\@gls@hyp@opt\@glspl}

```

Defined the un-starred form. Need to determine if there is a final optional argument

```

3666 \newcommand*{\@glspl}[2][{}]{%
3667   \new@ifnextchar[{\@glspl@{#1}{#2}}{\@glspl@{#1}{#2}[]}%
3668 }

```

`\@glspl@` Read in the final optional argument:

```

3669 \def\@glspl@#1#2[#3]{%
3670   \glsdoifexists{#2}%
3671   {%
3672     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper

3673     \let\glsifplural\@firstoftwo
3674     \let\glsapscase\@firstofthree
3675     \let\glscustomtext\@empty
3676     \def\glsinsert{#3}%

```

Determine what the link text should be (this is stored in `\@glo@text`) Note that `\@gls@link` sets `\glstype`.

```

3677   \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%

```

Call `\@gls@link`. If footnote package option has been used and the glossary type is `\acronymtype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```

3678   \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3679   \ifKV@glslink@local
3680     \glslocalunset{#2}%
3681   \else
3682     \glsunset{#2}%
3683   \fi
3684 }%

3685 \glspostlinkhook
3686 }

```

`\Glspl` behaves in the same way as `\glspl`, except that the first letter of the link text is converted to uppercase (as with `\Gls`, if the first letter has an accent, it will need to be grouped).

`\Glspl`

```

3687 \newrobustcmd*{\Glspl}{\@gls@hyp@opt\@Glspl}

```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3688 \newcommand*{\@Glspl}[2] [] {%
3689   \new@ifnextchar[{\@Glspl@{#1}{#2}}{\@Glspl@{#1}{#2} []}%
3690 }
```

`\@Glspl@` Read in the final optional argument:

```
3691 \def\@Glspl@#1#2[#3] {%
3692   \glsdoifexists{#2}%
3693   {%
3694     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3695     \let\glsifplural\@firstoftwo
3696     \let\glsupcase\@secondofthree
3697     \let\glscustomtext\@empty
3698     \def\glsinsert{#3}%
```

Determine what the link text should be (this is stored in `\@glo@text`). This needs to be expanded so that the `\@glo@text` can be passed to `\xmakefirstuc`. Note that `\@gls@link` sets `\glstyle`.

```
3699   \def\@glo@text{\csname gls@\glstyle @entryfmt\endcsname}%
```

Call `\@gls@link`. If footnote package option has been used and the glossary type is `\acronymtype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```
3700   \@gls@link[#1]{#2}{\@glo@text}%
```

Indicate that this entry has now been used

```
3701   \ifKV@glslink@local
3702     \glslocalunset{#2}%
3703   \else
3704     \glsunset{#2}%
3705   \fi
3706 }%
3707 \glspostlinkhook
3708 }
```

`\GLSp1` behaves like `\glspl` except that all the link text is converted to uppercase.

`\GLSp1`

```
3709 \newrobustcmd*{\GLSp1}{\@gls@hyp@opt\@GLSp1}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3710 \newcommand*{\@GLSp1}[2] [] {%
3711   \new@ifnextchar[{\@GLSp1@{#1}{#2}}{\@GLSp1@{#1}{#2} []}%
3712 }
```

`\@GLSp1` Read in the final optional argument:

```
3713 \def\@GLSp1@#1#2[#3] {%
3714   \glsdoifexists{#2}%
3715   {%
3716     \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
```

```

3717 \let\glsifplural\@firstoftwo
3718 \let\glscapscase\@thirdofthree
3719 \let\glscustomtext\@empty
3720 \def\glsinsert{#3}%

```

Determine what the link text should be (this is stored in \@glo@text) Note that \@gls@link sets \glstype.

```

3721 \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%

```

Call \@gls@link. If footnote package option has been used and the glossary type is \acronymtype, suppress hyperlink for first use. Likewise if the hyperfirst=false package option is used.

```

3722 \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```

3723 \ifKV@glslink@local
3724 \glslocalunset{#2}%
3725 \else
3726 \glsunset{#2}%
3727 \fi
3728 }%

```

```

3729 \glspostlinkhook
3730 }

```

`\glsdisp` `\glsdisp[<options>]{<label>}{<text>}` This is like `\gls` except that the link text is provided. This differs from `\glslink` in that it uses `\glsdisplay` or `\glsdisplayfirst` and unsets the first use flag.

First determine if we are using the starred form:

```

3731 \newrobustcmd*{\glsdisp}{\@gls@hyp@opt\@glsdisp}

```

Defined the un-starred form.

`\@glsdisp`

```

3732 \newcommand*{\@glsdisp}[3][{}]{%
3733 \glsdoifexists{#2}{%
3734 \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
3735 \let\glsifplural\@secondoftwo
3736 \let\glscapscase\@firstofthree
3737 \def\glscustomtext{#3}%
3738 \def\glsinsert{}%

```

Determine what the link text should be (this is stored in \@glo@text) Note that \@gls@link sets \glstype.

```

3739 \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%

```

Call \@gls@link. If footnote package option has been used and the glossary type is \acronymtype, suppress hyperlink for first use. Likewise if the hyperfirst=false package option is used.

```

3740 \@gls@link[#1]{#2}{\@glo@text}%

```

Indicate that this entry has now been used

```
3741 \ifKV@glslink@local
3742 \glsllocalunset{#2}%
3743 \else
3744 \glunset{#2}%
3745 \fi
3746 }%

3747 \glspostlinkhook
3748 }
```

checkfirsthyper Instead of just setting \do@gl@link@checkfirsthyper to \relax in \@gl@field@link, set it to \@gl@link@nocheckfirsthyper in case some other action needs to take place.

```
3749 \newcommand*{\@gl@link@nocheckfirsthyper}{}
```

@gl@field@link

```
3750 \newcommand{\@gl@field@link}[3]{%
3751 \glsoifexists{#2}%
3752 {%
3753 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
3754 \@gl@link[#1]{#2}{#3}%
3755 }%

3756 \glspostlinkhook
3757 }
```

\glstext behaves like \gls except it always uses the value given by the text key and it doesn't mark the entry as used.

\glstext

```
3758 \newrobustcmd*{\glstext}{\@gl@hyp@opt\@glstext}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3759 \newcommand*{\@glstext}[2][{}]{%
3760 \new@ifnextchar[{\@glstext@{#1}{#2}}{\@glstext@{#1}{#2}[]}}
```

Read in the final optional argument:

```
3761 \def\@glstext@#1#2[#3]{%
3762 \@gl@field@link{#1}{#2}{\glsentrytext{#2}#3}%
3763 }
```

\GLStext behaves like \glstext except the text is converted to uppercase.

\GLStext

```
3764 \newrobustcmd*{\GLStext}{\@gl@hyp@opt\@GLStext}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3765 \newcommand*{\@GLStext}[2][{}]{%
3766 \new@ifnextchar[{\@GLStext@{#1}{#2}}{\@GLStext@{#1}{#2}[]}}
```

Read in the final optional argument:

```
3767 \def\@GLStext@#1#2[#3]{%
3768   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glstentrytext{#2}#3}}%
3769 }
```

`\Glstext` behaves like `\glstext` except that the first letter of the text is converted to uppercase.

`\Glstext`

```
3770 \newrobustcmd*{\Glstext}{\@gls@hyp@opt\@GLstext}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3771 \newcommand*{\@GLstext}[2] [] {%
3772   \new@ifnextchar[{\@GLstext@{#1}{#2}}{\@GLstext@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3773 \def\@GLstext@#1#2[#3]{%
3774   \@gls@field@link{#1}{#2}{\Glstentrytext{#2}#3}%
3775 }
```

`\glsfirst` behaves like `\gls` except it always uses the value given by the first key and it doesn't mark the entry as used.

`\glsfirst`

```
3776 \newrobustcmd*{\glsfirst}{\@gls@hyp@opt\@glsfirst}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3777 \newcommand*{\@glsfirst}[2] [] {%
3778   \new@ifnextchar[{\@glsfirst@{#1}{#2}}{\@glsfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3779 \def\@glsfirst@#1#2[#3]{%
3780   \@gls@field@link{#1}{#2}{\glstentryfirst{#2}#3}%
3781 }
```

`\Glsfirst` behaves like `\glsfirst` except it displays the first letter in uppercase.

`\Glsfirst`

```
3782 \newrobustcmd*{\Glsfirst}{\@gls@hyp@opt\@Glsfirst}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3783 \newcommand*{\@Glsfirst}[2] [] {%
3784   \new@ifnextchar[{\@Glsfirst@{#1}{#2}}{\@Glsfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3785 \def\@Glsfirst@#1#2[#3]{%
3786   \@gls@field@link{#1}{#2}{\Glsentryfirst{#2}#3}%
3787 }
```

`\GLSfirst` behaves like `\Glsfirst` except it displays the text in uppercase.

`\GLSfirst`

```
3788 \newrobustcmd*{\GLSfirst}{\@gls@hyp@opt\@GLSfirst}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3789 \newcommand*{\@GLSfirst}[2] [] {%
3790   \new@ifnextchar[{\@GLSfirst@{#1}{#2}}{\@GLSfirst@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3791 \def\@GLSfirst@#1#2[#3] {%
3792   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirst{#2}#3}}%
3793 }
```

`\glsplural` behaves like `\gls` except it always uses the value given by the plural key and it doesn't mark the entry as used.

`\glsplural`

```
3794 \newrobustcmd*{\glsplural}{\@gls@hyp@opt\@glsplural}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3795 \newcommand*{\@glsplural}[2] [] {%
3796   \new@ifnextchar[{\@glsplural@{#1}{#2}}{\@glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3797 \def\@glsplural@#1#2[#3] {%
3798   \@gls@field@link{#1}{#2}{\glsentryplural{#2}#3}}%
3799 }
```

`\Glsplural` behaves like `\glsplural` except that the first letter is converted to uppercase.

`\Glsplural`

```
3800 \newrobustcmd*{\Glsplural}{\@gls@hyp@opt\@Glsplural}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3801 \newcommand*{\@Glsplural}[2] [] {%
3802   \new@ifnextchar[{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3803 \def\@Glsplural@#1#2[#3] {%
3804   \@gls@field@link{#1}{#2}{\Glsentryplural{#2}#3}}%
3805 }
```

`\GLSplural` behaves like `\glsplural` except that the text is converted to uppercase.

`\GLSplural`

```
3806 \newrobustcmd*{\GLSplural}{\@gls@hyp@opt\@GLSplural}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3807 \newcommand*{\@GLSplural}[2] [] {%
3808   \new@ifnextchar[{\@GLSplural@{#1}{#2}}{\@GLSplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3809 \def\@GLSplural@#1#2[#3] {%
3810   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryplural{#2}#3}}%
3811 }
```

`\glsfirstplural` behaves like `\gls` except it always uses the value given by the firstplural key and it doesn't mark the entry as used.

`\glsfirstplural`

```
3812 \newrobustcmd*{\glsfirstplural}{\@gls@hyp@opt\@glsfirstplural}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3813 \newcommand*{\@glsfirstplural}[2] [] {%
```

```
3814   \new@ifnextchar[{\@glsfirstplural@{#1}{#2}}{\@glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3815 \def\@glsfirstplural@#1#2[#3] {%
```

```
3816   \@gls@field@link{#1}{#2}{\glsentryfirstplural{#2}#3}%
```

```
3817 }
```

`\Glsfirstplural` behaves like `\glsfirstplural` except that the first letter is converted to uppercase.

`\Glsfirstplural`

```
3818 \newrobustcmd*{\Glsfirstplural}{\@gls@hyp@opt\@Glsfirstplural}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3819 \newcommand*{\@Glsfirstplural}[2] [] {%
```

```
3820   \new@ifnextchar[{\@Glsfirstplural@{#1}{#2}}{\@Glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3821 \def\@Glsfirstplural@#1#2[#3] {%
```

```
3822   \@gls@field@link{#1}{#2}{\Glsentryfirstplural{#2}#3}%
```

```
3823 }
```

`\GLSfirstplural` behaves like `\glsfirstplural` except that the link text is converted to uppercase.

`\GLSfirstplural`

```
3824 \newrobustcmd*{\GLSfirstplural}{\@gls@hyp@opt\@GLSfirstplural}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3825 \newcommand*{\@GLSfirstplural}[2] [] {%
```

```
3826   \new@ifnextchar[{\@GLSfirstplural@{#1}{#2}}{\@GLSfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3827 \def\@GLSfirstplural@#1#2[#3] {%
```

```
3828   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryfirstplural{#2}#3}}%
```

```
3829 }
```

`\glsname` behaves like `\gls` except it always uses the value given by the name key and it doesn't mark the entry as used.

`\glsname`

```
3830 \newrobustcmd*{\glsname}{\@gls@hyp@opt\@glsname}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3831 \newcommand*{\@glsname}[2] [] {%
```

```
3832   \new@ifnextchar[{\@glsname@{#1}{#2}}{\@glsname@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3833 \def\@glsname@#1#2[#3]{%
3834   \@gls@field@link{#1}{#2}{\glsentryname{#2}#3}%
3835 }
```

\Glsname behaves like \glsname except that the first letter is converted to uppercase.

\Glsname

```
3836 \newrobustcmd*{\Glsname}{\@gls@hyp@opt\@Glsname}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3837 \newcommand*{\@Glsname}[2][\%]
3838   \new@ifnextchar[{\@Glsname@{#1}{#2}}{\@Glsname@{#1}{#2}[]}}
```

Read in the final optional argument:

```
3839 \def\@Glsname@#1#2[#3]{%
3840   \@gls@field@link{#1}{#2}{\glsentryname{#2}#3}%
3841 }
```

\GLSname behaves like \glsname except that the link text is converted to uppercase.

\GLSname

```
3842 \newrobustcmd*{\GLSname}{\@gls@hyp@opt\@GLSname}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3843 \newcommand*{\@GLSname}[2][\%]
3844   \new@ifnextchar[{\@GLSname@{#1}{#2}}{\@GLSname@{#1}{#2}[]}]
```

Read in the final optional argument:

```
3845 \def\@GLSname@#1#2[#3]{%
3846   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryname{#2}#3}}%
3847 }
```

\glsdesc behaves like \gls except it always uses the value given by the description key and it doesn't mark the entry as used.

\glsdesc

```
3848 \newrobustcmd*{\glsdesc}{\@gls@hyp@opt\@glsdesc}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3849 \newcommand*{\@glsdesc}[2][\%]
3850   \new@ifnextchar[{\@glsdesc@{#1}{#2}}{\@glsdesc@{#1}{#2}[]}]
```

Read in the final optional argument:

```
3851 \def\@glsdesc@#1#2[#3]{%
3852   \@gls@field@link{#1}{#2}{\glsentrydesc{#2}#3}%
3853 }
```

\Glsdesc behaves like \glsdesc except that the first letter is converted to uppercase.

\Glsdesc

```
3854 \newrobustcmd*{\Glsdesc}{\@gls@hyp@opt\@Glsdesc}
```


Define the un-starred form. Need to determine if there is a final optional argument

```
3855 \newcommand*{\@GLSdesc}[2] [] {%  
3856   \new@ifnextchar [{\@GLSdesc@{#1}{#2}}{\@GLSdesc@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3857 \def\@GLSdesc@#1#2[#3] {%  
3858   \@gls@field@link{#1}{#2}{\@Glsentrydesc{#2}#3}%  
3859 }
```

\GLSdesc behaves like \glsdesc except that the link text is converted to uppercase.

\GLSdesc

```
3860 \newrobustcmd*{\GLSdesc}{\@gls@hyp@opt\@GLSdesc}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3861 \newcommand*{\@GLSdesc}[2] [] {%  
3862   \new@ifnextchar [{\@GLSdesc@{#1}{#2}}{\@GLSdesc@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3863 \def\@GLSdesc@#1#2[#3] {%  
3864   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\@Glsentrydesc{#2}#3}}%  
3865 }
```

\glsdescplural behaves like \gls except it always uses the value given by the description-plural key and it doesn't mark the entry as used.

\glsdescplural

```
3866 \newrobustcmd*{\glsdescplural}{\@gls@hyp@opt\@glsdescplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3867 \newcommand*{\@glsdescplural}[2] [] {%  
3868   \new@ifnextchar [{\@glsdescplural@{#1}{#2}}{\@glsdescplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3869 \def\@glsdescplural@#1#2[#3] {%  
3870   \@gls@field@link{#1}{#2}{\@Glsentrydescplural{#2}#3}%  
3871 }
```

\Glsdescplural behaves like \glsdescplural except that the first letter is converted to uppercase.

\Glsdescplural

```
3872 \newrobustcmd*{\Glsdescplural}{\@gls@hyp@opt\@Glsdescplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3873 \newcommand*{\@Glsdescplural}[2] [] {%  
3874   \new@ifnextchar [{\@Glsdescplural@{#1}{#2}}{\@Glsdescplural@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3875 \def\@Glsdescplural@#1#2[#3] {%  
3876   \@gls@field@link{#1}{#2}{\@Glsentrydescplural{#2}#3}%  
3877 }
```

`\GLSdescplural` behaves like `\glsdescplural` except that the link text is converted to uppercase.

`\GLSdescplural`

```
3878 \newrobustcmd*{\GLSdescplural}{\@gls@hyp@opt\@GLSdescplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3879 \newcommand*{\@GLSdescplural}[2] [] {%
3880   \new@ifnextchar[{\@GLSdescplural@{#1}{#2}}{\@GLSdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3881 \def\@GLSdescplural@#1#2[#3] {%
3882   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrydescplural{#2}#3}}%
3883 }
```

`\glsymbol` behaves like `\gls` except it always uses the value given by the symbol key and it doesn't mark the entry as used.

`\glsymbol`

```
3884 \newrobustcmd*{\glsymbol}{\@gls@hyp@opt\@glsymbol}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3885 \newcommand*{\@glsymbol}[2] [] {%
3886   \new@ifnextchar[{\@glsymbol@{#1}{#2}}{\@glsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3887 \def\@glsymbol@#1#2[#3] {%
3888   \@gls@field@link{#1}{#2}{\glsentrysymbol{#2}#3}%
3889 }
```

`\Glsymbol` behaves like `\glsymbol` except that the first letter is converted to uppercase.

`\Glsymbol`

```
3890 \newrobustcmd*{\Glsymbol}{\@gls@hyp@opt\@Glsymbol}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3891 \newcommand*{\@Glsymbol}[2] [] {%
3892   \new@ifnextchar[{\@Glsymbol@{#1}{#2}}{\@Glsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3893 \def\@Glsymbol@#1#2[#3] {%
3894   \@gls@field@link{#1}{#2}{\Glsentrysymbol{#2}#3}%
3895 }
```

`\GLSsymbol` behaves like `\glsymbol` except that the link text is converted to uppercase.

`\GLSsymbol`

```
3896 \newrobustcmd*{\GLSsymbol}{\@gls@hyp@opt\@GLSsymbol}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3897 \newcommand*{\@GLSsymbol}[2] [] {%
3898   \new@ifnextchar[{\@GLSsymbol@{#1}{#2}}{\@GLSsymbol@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3899 \def\@GLSsymbol@#1#2[#3]{%
3900   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrysymbol{#2}#3}}%
3901 }
```

`\glsymbolplural` behaves like `\gls` except it always uses the value given by the symbolplural key and it doesn't mark the entry as used.

`glsymbolplural`

```
3902 \newrobustcmd*{\glsymbolplural}{\@gls@hyp@opt\@glsymbolplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3903 \newcommand*{\@glsymbolplural}[2][ ]{%
3904   \new@ifnextchar[{\@glsymbolplural@{#1}{#2}}{\@glsymbolplural@{#1}{#2}[ ]}]}
```

Read in the final optional argument:

```
3905 \def\@glsymbolplural@#1#2[#3]{%
3906   \@gls@field@link{#1}{#2}{\glsentrysymbolplural{#2}#3}%
3907 }
```

`\Glsymbolplural` behaves like `\glsymbolplural` except that the first letter is converted to uppercase.

`Glsymbolplural`

```
3908 \newrobustcmd*{\Glsymbolplural}{\@gls@hyp@opt\@Glsymbolplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3909 \newcommand*{\@Glsymbolplural}[2][ ]{%
3910   \new@ifnextchar[{\@Glsymbolplural@{#1}{#2}}{\@Glsymbolplural@{#1}{#2}[ ]}]}
```

Read in the final optional argument:

```
3911 \def\@Glsymbolplural@#1#2[#3]{%
3912   \@gls@field@link{#1}{#2}{\Glsentrysymbolplural{#2}#3}%
3913 }
```

`\GLSsymbolplural` behaves like `\glsymbolplural` except that the link text is converted to uppercase.

`GLSsymbolplural`

```
3914 \newrobustcmd*{\GLSsymbolplural}{\@gls@hyp@opt\@GLSsymbolplural}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3915 \newcommand*{\@GLSsymbolplural}[2][ ]{%
3916   \new@ifnextchar[{\@GLSsymbolplural@{#1}{#2}}{\@GLSsymbolplural@{#1}{#2}[ ]}]}
```

Read in the final optional argument:

```
3917 \def\@GLSsymbolplural@#1#2[#3]{%
3918   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentrysymbolplural{#2}#3}}%
3919 }
```

`\glsuseri` behaves like `\gls` except it always uses the value given by the `user1` key and it doesn't mark the entry as used.

\glsuseri

```
3920 \newrobustcmd*{\glsuseri}{\@gls@hyp@opt\@glsuseri}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3921 \newcommand*{\@glsuseri}[2] [] {%
```

```
3922   \new@ifnextchar[{\@glsuseri@{#1}{#2}}{\@glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3923 \def\@glsuseri@#1#2[#3]{%
```

```
3924   \@gls@field@link{#1}{#2}{\glsentryuseri{#2}#3}%
```

```
3925 }
```

\Glsuseri behaves like \glsuseri except that the first letter is converted to uppercase.

\Glsuseri

```
3926 \newrobustcmd*{\Glsuseri}{\@gls@hyp@opt\@Glsuseri}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3927 \newcommand*{\@Glsuseri}[2] [] {%
```

```
3928   \new@ifnextchar[{\@Glsuseri@{#1}{#2}}{\@Glsuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3929 \def\@Glsuseri@#1#2[#3]{%
```

```
3930   \@gls@field@link{#1}{#2}{\Glsentryuseri{#2}#3}%
```

```
3931 }
```

\GLSuseri behaves like \glsuseri except that the link text is converted to uppercase.

\GLSuseri

```
3932 \newrobustcmd*{\GLSuseri}{\@gls@hyp@opt\@GLSuseri}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3933 \newcommand*{\@GLSuseri}[2] [] {%
```

```
3934   \new@ifnextchar[{\@GLSuseri@{#1}{#2}}{\@GLSuseri@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3935 \def\@GLSuseri@#1#2[#3]{%
```

```
3936   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseri{#2}#3}}%
```

```
3937 }
```

\glsuserii behaves like \gls except it always uses the value given by the user2 key and it doesn't mark the entry as used.

\glsuserii

```
3938 \newrobustcmd*{\glsuserii}{\@gls@hyp@opt\@glsuserii}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3939 \newcommand*{\@glsuserii}[2] [] {%
```

```
3940   \new@ifnextchar[{\@glsuserii@{#1}{#2}}{\@glsuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3941 \def\@glsuserii@#1#2[#3]{%
```

```
3942   \@gls@field@link{#1}{#2}{\glsentryuserii{#2}#3}%
```

```
3943 }
```

\Glsuserii behaves like \glsuserii except that the first letter is converted to uppercase.

\Glsuserii

```
3944 \newrobustcmd*{\Glsuserii}{\@gls@hyp@opt\@Glsuserii}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3945 \newcommand*{\@Glsuserii}[2] [] {%
```

```
3946   \new@ifnextchar[{\@Glsuserii@{#1}{#2}}{\@Glsuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3947 \def\@Glsuserii@#1#2[#3] {%
```

```
3948   \@gls@field@link{#1}{#2}{\Glsentryuserii{#2}#3}%
```

```
3949 }
```

\GLSuserii behaves like \glsuserii except that the link text is converted to uppercase.

\GLSuserii

```
3950 \newrobustcmd*{\GLSuserii}{\@gls@hyp@opt\@GLSuserii}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
3951 \newcommand*{\@GLSuserii}[2] [] {%
```

```
3952   \new@ifnextchar[{\@GLSuserii@{#1}{#2}}{\@GLSuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3953 \def\@GLSuserii@#1#2[#3] {%
```

```
3954   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryuserii{#2}#3}}%
```

```
3955 }
```

\glsuseriii behaves like \gls except it always uses the value given by the user3 key and it doesn't mark the entry as used.

\glsuseriii

```
3956 \newrobustcmd*{\glsuseriii}{\@gls@hyp@opt\@glsuseriii}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3957 \newcommand*{\@glsuseriii}[2] [] {%
```

```
3958   \new@ifnextchar[{\@glsuseriii@{#1}{#2}}{\@glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3959 \def\@glsuseriii@#1#2[#3] {%
```

```
3960   \@gls@field@link{#1}{#2}{\glsentryuseriii{#2}#3}%
```

```
3961 }
```

\Glsuseriii behaves like \glsuseriii except that the first letter is converted to uppercase.

\Glsuseriii

```
3962 \newrobustcmd*{\Glsuseriii}{\@gls@hyp@opt\@Glsuseriii}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3963 \newcommand*{\@Glsuseriii}[2] [] {%
```

```
3964   \new@ifnextchar[{\@Glsuseriii@{#1}{#2}}{\@Glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
3965 \def\@Glsuseriii@#1#2[#3]{%
3966   \@gls@field@link{#1}{#2}{\Glsentryuseriii{#2}#3}%
3967 }
```

\Glsuseriii behaves like \glsuseriii except that the link text is converted to uppercase.

\Glsuseriii

```
3968 \newrobustcmd*{\Glsuseriii}{\@gls@hyp@opt\@Glsuseriii}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3969 \newcommand*{\@Glsuseriii}[2][\@Glsuseriii@#1]{%
3970   \new@ifnextchar[\@Glsuseriii@{#1}{#2}]{\@Glsuseriii@{#1}{#2}[]}}
```

Read in the final optional argument:

```
3971 \def\@Glsuseriii@#1#2[#3]{%
3972   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\Glsentryuseriii{#2}#3}}%
3973 }
```

\glsuseriv behaves like \gls except it always uses the value given by the user4 key and it doesn't mark the entry as used.

\glsuseriv

```
3974 \newrobustcmd*{\glsuseriv}{\@gls@hyp@opt\@glsuseriv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3975 \newcommand*{\@glsuseriv}[2][\@glsuseriv@#1]{%
3976   \new@ifnextchar[\@glsuseriv@{#1}{#2}]{\@glsuseriv@{#1}{#2}[]}}
```

Read in the final optional argument:

```
3977 \def\@glsuseriv@#1#2[#3]{%
3978   \@gls@field@link{#1}{#2}{\Glsentryuseriv{#2}#3}%
3979 }
```

\Glsuseriv behaves like \glsuseriv except that the first letter is converted to uppercase.

\Glsuseriv

```
3980 \newrobustcmd*{\Glsuseriv}{\@gls@hyp@opt\@Glsuseriv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3981 \newcommand*{\@Glsuseriv}[2][\@Glsuseriv@#1]{%
3982   \new@ifnextchar[\@Glsuseriv@{#1}{#2}]{\@Glsuseriv@{#1}{#2}[]}}
```

Read in the final optional argument:

```
3983 \def\@Glsuseriv@#1#2[#3]{%
3984   \@gls@field@link{#1}{#2}{\Glsentryuseriv{#2}#3}%
3985 }
```

\Glsuseriv behaves like \glsuseriv except that the link text is converted to uppercase.

\Glsuseriv

```
3986 \newrobustcmd*{\Glsuseriv}{\@gls@hyp@opt\@Glsuseriv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3987 \newcommand*{\@GLSuseriv}[2] [] {%  
3988   \new@ifnextchar [{\@GLSuseriv@{#1}{#2}}{\@GLSuseriv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3989 \def\@GLSuseriv@#1#2[#3] {%  
3990   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuseriv{#2}#3}}%  
3991 }
```

\glsuserv behaves like \gls except it always uses the value given by the user5 key and it doesn't mark the entry as used.

\glsuserv

```
3992 \newrobustcmd*{\glsuserv}{\@gls@hyp@opt\@glsuserv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3993 \newcommand*{\@glsuserv}[2] [] {%  
3994   \new@ifnextchar [{\@glsuserv@{#1}{#2}}{\@glsuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
3995 \def\@glsuserv@#1#2[#3] {%  
3996   \@gls@field@link{#1}{#2}{\glsentryuserv{#2}#3}}%  
3997 }
```

\Glsuserv behaves like \glsuserv except that the first letter is converted to uppercase.

\Glsuserv

```
3998 \newrobustcmd*{\Glsuserv}{\@gls@hyp@opt\@Glsuserv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
3999 \newcommand*{\@Glsuserv}[2] [] {%  
4000   \new@ifnextchar [{\@Glsuserv@{#1}{#2}}{\@Glsuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4001 \def\@Glsuserv@#1#2[#3] {%  
4002   \@gls@field@link{#1}{#2}{\Glsentryuserv{#2}#3}}%  
4003 }
```

\GLSuserv behaves like \glsuserv except that the link text is converted to uppercase.

\GLSuserv

```
4004 \newrobustcmd*{\GLSuserv}{\@gls@hyp@opt\@GLSuserv}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4005 \newcommand*{\@GLSuserv}[2] [] {%  
4006   \new@ifnextchar [{\@GLSuserv@{#1}{#2}}{\@GLSuserv@{#1}{#2} [] }}
```

Read in the final optional argument:

```
4007 \def\@GLSuserv@#1#2[#3] {%  
4008   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuserv{#2}#3}}%  
4009 }
```

\glsuservi behaves like \gls except it always uses the value given by the user6 key and it doesn't mark the entry as used.

\glsuservi

```
4010 \newrobustcmd*{\glsuservi}{\@gls@hyp@opt\@glsuservi}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
4011 \newcommand*{\@glsuservi}[2] [] {%
```

```
4012   \new@ifnextchar[{\@glsuservi@{#1}{#2}}{\@glsuservi@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4013 \def\@glsuservi@#1#2[#3] {%
```

```
4014   \@gls@field@link{#1}{#2}{\glsentryuservi{#2}#3}%
```

```
4015 }
```

\Glsuservi behaves like \glsuservi except that the first letter is converted to uppercase.

\Glsuservi

```
4016 \newrobustcmd*{\Glsuservi}{\@gls@hyp@opt\@Glsuservi}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
4017 \newcommand*{\@Glsuservi}[2] [] {%
```

```
4018   \new@ifnextchar[{\@Glsuservi@{#1}{#2}}{\@Glsuservi@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4019 \def\@Glsuservi@#1#2[#3] {%
```

```
4020   \@gls@field@link{#1}{#2}{\Glsentryuservi{#2}#3}%
```

```
4021 }
```

\GLSuservi behaves like \glsuservi except that the link text is converted to uppercase.

\GLSuservi

```
4022 \newrobustcmd*{\GLSuservi}{\@gls@hyp@opt\@GLSuservi}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4023 \newcommand*{\@GLSuservi}[2] [] {%
```

```
4024   \new@ifnextchar[{\@GLSuservi@{#1}{#2}}{\@GLSuservi@{#1}{#2} []}]}
```

Read in the final optional argument:

```
4025 \def\@GLSuservi@#1#2[#3] {%
```

```
4026   \@gls@field@link{#1}{#2}{\mfirstucMakeUppercase{\glsentryuservi{#2}#3}}%
```

```
4027 }
```

Now deal with acronym related keys. First the short form:

\acrshort

```
4028 \newrobustcmd*{\acrshort}{\@gls@hyp@opt\@ns@acrshort}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4029 \newcommand*{\@ns@acrshort}[2] [] {%
```

```
4030   \new@ifnextchar[{\@acrshort{#1}{#2}}{\@acrshort{#1}{#2} []}]%
```

```
4031 }
```

Read in the final optional argument:

```
4032 \def\@acrshort#1#2[#3] {%
```

```
4033   \glsdoifexists{#2}%
```

```
4034   {%
```



```
4035 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
```

```
4036 \let\gl@sifplural\@secondoftwo
```

```
4037 \let\gl@scapscase\@firstofthree
```

```
4038 \let\gl@insert\@empty
```

```
4039 \def\glscustomtext{%
```

```
4040 \acronymfont{\gl@sentryshort{#2}}#3%
```

```
4041 }%
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
4042 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
```

```
4043 }%
```

```
4044 \glspostlinkhook
```

```
4045 }
```

\Acrshort

```
4046 \newrobustcmd*{\Acrshort}{\@gl@hyp@opt\@ns@Acrshort}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4047 \newcommand*{\ns@Acrshort}[2][{}]{%
```

```
4048 \new@ifnextchar[{\@Acrshort{#1}{#2}}{\@Acrshort{#1}{#2}[]}%
```

```
4049 }
```

Read in the final optional argument:

```
4050 \def\@Acrshort#1#2[#3]{%
```

```
4051 \gl@sdoifexists{#2}%
```

```
4052 {%
```

```
4053 \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
```

```
4054 \def\glslabel{#2}%
```

```
4055 \let\gl@sifplural\@secondoftwo
```

```
4056 \let\gl@scapscase\@secondofthree
```

```
4057 \let\gl@insert\@empty
```

```
4058 \def\glscustomtext{%
```

```
4059 \acronymfont{\Glsentryshort{#2}}#3%
```

```
4060 }%
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
4061 \@gl@link[#1]{#2}{\csname gls@glstype @entryfmt\endcsname}%
```

```
4062 }%
```

```
4063 \glspostlinkhook
```

```
4064 }
```

\ACRshort

```
4065 \newrobustcmd*{\ACRshort}{\@gl@hyp@opt\@ns@ACRshort}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4066 \newcommand*{\ns@ACRshort}[2][\%
4067   \new@ifnextchar[\@ACRshort{#1}{#2}]{\@ACRshort{#1}{#2}[]}%
4068 }
```

Read in the final optional argument:

```
4069 \def\@ACRshort#1#2[#3]{%
4070   \glstoifexists{#2}%
4071   {%
4072     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4073     \def\glslabel{#2}%
4074     \let\gl@ifplural\@secondoftwo
4075     \let\glscapscase\@thirdofthree
4076     \let\glinsert\@empty
4077     \def\glscustomtext{%
4078       \mfirstucMakeUppercase{\acronymfont{\glentryshort{#2}}#3}%
4079     }%
```

Call \@gl@link Note that \@gl@link sets \glstype.

```
4080   \@gl@link[#1]{#2}{\csname gls\glstype @entryfmt\endcsname}%
4081   }%
4082   \glspostlinkhook
4083 }
```

Short plural:

\acrshortpl

```
4084 \newrobustcmd*{\acrshortpl}{\@gl@hyp@opt\ns@acrshortpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4085 \newcommand*{\ns@acrshortpl}[2][\%
4086   \new@ifnextchar[\@acrshortpl{#1}{#2}]{\@acrshortpl{#1}{#2}[]}%
4087 }
```

Read in the final optional argument:

```
4088 \def\@acrshortpl#1#2[#3]{%
4089   \glstoifexists{#2}%
4090   {%
4091     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4092     \def\glslabel{#2}%
4093     \let\gl@ifplural\@firstoftwo
4094     \let\glscapscase\@firstofthree
4095     \let\glinsert\@empty
4096     \def\glscustomtext{%
4097       \acronymfont{\glentryshortpl{#2}}#3%
4098     }%
```

Call \@gls@link Note that \@gls@link sets \glsstyle.

```
4099   \@gls@link[#1]{#2}{\csname gls@\glsstyle @entryfmt\endcsname}%  
4100   }%  
  
4101   \glspostlinkhook  
4102 }
```

\Acrshortpl

```
4103 \newrobustcmd*{\Acrshortpl}{\@gls@hyp@opt\ns@Acrshortpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4104 \newcommand*{\ns@Acrshortpl}[2] [] {%  
4105   \new@ifnextchar[{\@Acrshortpl{#1}{#2}}{\@Acrshortpl{#1}{#2} []}%  
4106 }
```

Read in the final optional argument:

```
4107 \def\@Acrshortpl#1#2[#3] {%  
4108   \glsdoifexists{#2}%  
4109   {%  
  
4110     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper  
  
4111     \def\glslabel{#2}%  
4112     \let\glsifplural\@firstoftwo  
4113     \let\glsescapescase\@secondofthree  
4114     \let\glsinsert\@empty  
4115     \def\glscustomtext{%  
4116       \acronymfont{\Glsentryshortpl{#2}}#3%  
4117     }%
```

Call \@gls@link Note that \@gls@link sets \glsstyle.

```
4118   \@gls@link[#1]{#2}{\csname gls@\glsstyle @entryfmt\endcsname}%  
4119   }%  
  
4120   \glspostlinkhook  
4121 }
```

\ACRshortpl

```
4122 \newrobustcmd*{\ACRshortpl}{\@gls@hyp@opt\ns@ACRshortpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4123 \newcommand*{\ns@ACRshortpl}[2] [] {%  
4124   \new@ifnextchar[{\@ACRshortpl{#1}{#2}}{\@ACRshortpl{#1}{#2} []}%  
4125 }
```

Read in the final optional argument:

```
4126 \def\@ACRshortpl#1#2[#3] {%  
4127   \glsdoifexists{#2}%  
4128   {%  
  
4129     \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
```

```

4130 \def\glslabel{#2}%
4131 \let\glsifplural\@firstoftwo
4132 \let\glsupcase\@thirdofthree
4133 \let\glsinsert\@empty
4134 \def\glscustomtext{%
4135     \mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{#2}}#3}%
4136 }%

```

Call \@gls@link Note that \@gls@link sets \glsstyle.

```

4137 \@gls@link[#1]{#2}{\csname gls@\glsstyle @entryfmt\endcsname}%
4138 }%

4139 \glspostlinkhook
4140 }

```

\acrlong

```

4141 \newrobustcmd*{\acrlong}{\@gls@hyp@opt\ns@acrlong}

```

Define the un-starred form. Need to determine if there is a final optional argument

```

4142 \newcommand*{\ns@acrlong}[2] [] {%
4143     \new@ifnextchar[{\@acrlong{#1}{#2}}{\@acrlong{#1}{#2} []}%
4144 }

```

Read in the final optional argument:

```

4145 \def\@acrlong#1#2[#3] {%
4146     \glsdoifexists{#2}%
4147     {%
4148         \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

4149         \def\glslabel{#2}%
4150         \let\glsifplural\@secondoftwo
4151         \let\glsupcase\@firstofthree
4152         \let\glsinsert\@empty

```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4153     \def\glscustomtext{%
4154         \glsentrylong{#2}#3%
4155     }%

```

Call \@gls@link Note that \@gls@link sets \glsstyle.

```

4156 \@gls@link[#1]{#2}{\csname gls@\glsstyle @entryfmt\endcsname}%
4157 }%

4158 \glspostlinkhook
4159 }

```

\Acrlong

```

4160 \newrobustcmd*{\Acrlong}{\@gls@hyp@opt\ns@Acrlong}

```

Define the un-starred form. Need to determine if there is a final optional argument

```
4161 \newcommand*{\ns@Acrlong}[2][\%]
4162   \new@ifnextchar[{\@Acrlong{#1}{#2}}{\@Acrlong{#1}{#2}[]}%
4163 }
```

Read in the final optional argument:

```
4164 \def\@Acrlong#1#2[#3]{%
4165   \glsdoifexists{#2}%
4166   {%
4167     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4168     \def\glslabel{#2}%
4169     \let\glsifplural\@secondoftwo
4170     \let\glscapscase\@secondofthree
4171     \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
4172   \def\glscustomtext{%
4173     \Glsentrylong{#2}#3%
4174   }%
```

Call \@gl@link. Note that \@gl@link sets \glstype.

```
4175   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4176   }%
4177   \glspostlinkhook
4178 }
```

\ACRlong

```
4179 \newrobustcmd*{\ACRlong}{\@gl@hyp@opt\ns@ACRlong}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4180 \newcommand*{\ns@ACRlong}[2][\%]
4181   \new@ifnextchar[{\@ACRlong{#1}{#2}}{\@ACRlong{#1}{#2}[]}%
4182 }
```

Read in the final optional argument:

```
4183 \def\@ACRlong#1#2[#3]{%
4184   \glsdoifexists{#2}%
4185   {%
4186     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4187     \def\glslabel{#2}%
4188     \let\glsifplural\@secondoftwo
4189     \let\glscapscase\@thirdofthree
4190     \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
4191 \def\glscustomtext{%
4192 \mfirstucMakeUppercase{\glsentrylong{#2}#3}%
4193 }%
```

Call \@gls@link. Note that \@gls@link sets \glstype.

```
4194 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4195 }%

4196 \glspostlinkhook
4197 }
```

Short plural:

\acrlongpl

```
4198 \newrobustcmd*{\acrlongpl}{\@gls@hyp@opt\@ns@acrlongpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4199 \newcommand*{\ns@acrlongpl}[2][\%
4200 \new@ifnextchar[\@acrlongpl{#1}{#2}]{\@acrlongpl{#1}{#2}[]}%
4201 }
```

Read in the final optional argument:

```
4202 \def\@acrlongpl#1#2[#3]{%
4203 \glsdoidexists{#2}%
4204 {%

4205 \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper

4206 \def\glslabel{#2}%
4207 \let\glsifplural\@firstoftwo
4208 \let\glscapscase\@firstofthree
4209 \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
4210 \def\glscustomtext{%
4211 \glsentrylongpl{#2}#3%
4212 }%
```

Call \@gls@link. Note that \@gls@link sets \glstype.

```
4213 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4214 }%

4215 \glspostlinkhook
4216 }
```

\Acrlongpl

```
4217 \newrobustcmd*{\Acrlongpl}{\@gls@hyp@opt\@ns@Acrlongpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4218 \newcommand*{\ns@Acrlongpl}[2][\%
4219   \new@ifnextchar[\@Acrlongpl{#1}{#2}]{\@Acrlongpl{#1}{#2}[]}%
4220 }
```

Read in the final optional argument:

```
4221 \def\@Acrlongpl#1#2[#3]{%
4222   \glsdoifexists{#2}%
4223   {%
4224     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4225     \def\glslabel{#2}%
4226     \let\glsifplural\@firstoftwo
4227     \let\glscapscase\@secondofthree
4228     \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```
4229   \def\glscustomtext{%
4230     \Glsentrylongpl{#2}#3%
4231   }%
```

Call \@gl@link. Note that \@gl@link sets \glstype.

```
4232   \@gl@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4233   }%
4234   \glspostlinkhook
4235 }
```

\ACRlongpl

```
4236 \newrobustcmd*{\ACRlongpl}{\@gl@hyp@opt\ns@ACRlongpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
4237 \newcommand*{\ns@ACRlongpl}[2][\%
4238   \new@ifnextchar[\@ACRlongpl{#1}{#2}]{\@ACRlongpl{#1}{#2}[]}%
4239 }
```

Read in the final optional argument:

```
4240 \def\@ACRlongpl#1#2[#3]{%
4241   \glsdoifexists{#2}%
4242   {%
4243     \let\do@gl@link@checkfirsthyper\@gl@link@nocheckfirsthyper
4244     \def\glslabel{#2}%
4245     \let\glsifplural\@firstoftwo
4246     \let\glscapscase\@thirdofthree
4247     \let\glsinsert\@empty
```

Bug fix v4.02 removed \acronymfont from \glscustomtext (\acronymfont only designed for short form).

```

4248 \def\glscustomtext{%
4249 \mfirstucMakeUppercase{\glsenentrylongpl{#2}#3}%
4250 }%

Call \@gls@link. Note that \@gls@link sets \glstype.
4251 \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
4252 }%

4253 \glspostlinkhook
4254 }

```

Displaying entry details without adding information to the glossary

These commands merely display entry information without adding entries in the associated file or having hyperlinks.

`\gls@entry@field` Generic version.

`\@gls@entry@field{<label>}{<field>}`

```

4255 \newcommand*{\@gls@entry@field}[2]{%
4256 \csname glo@\glsdetoklabel{#1}@#2\endcsname
4257 }

```

`\glsletentryfield` `\glsletentryfield{<cs>}{<label>}{<field>}`

```

4258 \newcommand*{\glsletentryfield}[3]{%
4259 \letcs{#1}{glo@\glsdetoklabel{#2}@#3}%
4260 }

```

`\Gls@entry@field` Generic first letter uppercase version.

`\@Gls@entry@field{<label>}{<field>}`

```

4261 \newcommand*{\@Gls@entry@field}[2]{%
4262 \glsdoifexistsordo{#1}%
4263 {%
4264 \letcs{\@glo@text}{glo@\glsdetoklabel{#1}@#2}%
4265 \ifdef\@glo@text
4266 {%
4267 \xmakefirstuc{\@glo@text}%
4268 }%
4269 }%
4270 ??\PackageError{glossaries}{The field ‘#2’ doesn’t exist for glossary
4271 entry ‘\glsdetoklabel{#1}’}{Check you have correctly spelt the entry

```



```

4272     label and the field name}%
4273 }%
4274 }%
4275 {%
4276 ??%
4277 }%
4278 }

```

Get the entry name (as specified by the name key when the entry was defined). The argument is the label associated with the entry. Note that unless you used `name=false` in the `sanitize` package option you may get unexpected results if the name key contains any commands.

`\glsentryname`

```

4279 \newcommand*{\glsentryname}[1]{\@gls@entry@field{#1}{name}}

```

`\Glsentryname`

```

4280 \newrobustcmd*{\Glsentryname}[1]{%
4281   \@Gls@entryname{#1}%
4282 }

```

`\@Gls@entryname` This is a workaround in the event that the user defies the warning in the manual about not using `\Glsname` or `\Glsentryname` with acronyms. First the default behaviour:

```

4283 \newcommand*{\@Gls@entryname}[1]{%
4284   \@Gls@entry@field{#1}{name}%
4285 }

```

`\ls@acrentryname` Now the behaviour when `\setacronymstyle` is used:

```

4286 \newcommand*{\@Gls@acrentryname}[1]{%
4287   \ifglshaslong{#1}%
4288   {%
4289     \letcs\@glo@text{glo\@glsdetoklabel{#1}@name}%
4290     \expandafter\@gls@getbody\@glo@text{}\@nil
4291     \expandafter\ifx\@gls@body\glsentrylong\relax
4292     \expandafter\Glsentrylong\@gls@rest
4293   }else
4294     \expandafter\ifx\@gls@body\glsentryshort\relax
4295     \expandafter\Glsentryshort\@gls@rest
4296   }else
4297     \expandafter\ifx\@gls@body\acronymfont\relax

```

Temporarily make `\glsentryshort` behave like `\Glsentryshort`. (This is on the assumption that the argument of `\acronymfont` is `\glsentryshort{<label>}`, as that's the behaviour of the predefined acronym styles.) This is scoped to localise the effect of the assignment.

```

4298     {%
4299       \let\glsentryshort\Glsentryshort
4300       \@glo@text
4301     }%
4302   }else

```

```

4303      \xmakefirstuc{\@glo@text}%
4304      \fi
4305      \fi
4306      \fi
4307  }%
4308  {%

```

Not an acronym

```

4309      \@Gls@entry@field{#1}{name}%
4310  }%
4311 }

```

Get the entry description (as specified by the description when the entry was defined). The argument is the label associated with the entry. Note that unless you used `description=false` in the `sanitize` package option you may get unexpected results if the description key contained any commands.

`\glsentrydesc`

```

4312 \newcommand*{\glsentrydesc}[1]{\@gls@entry@field{#1}{desc}}

```

`\Glsentrydesc`

```

4313 \newrobustcmd*{\Glsentrydesc}[1]{%
4314   \@Gls@entry@field{#1}{desc}%
4315 }

```

Plural form:

`entrydescplural`

```

4316 \newcommand*{\glsentrydescplural}[1]{%
4317   \@gls@entry@field{#1}{descplural}%
4318 }

```

`entrydescplural`

```

4319 \newrobustcmd*{\Glsentrydescplural}[1]{%
4320   \@Gls@entry@field{#1}{descplural}%
4321 }

```

Get the entry text, as specified by the text key when the entry was defined. The argument is the label associated with the entry:

`\glsentrytext`

```

4322 \newcommand*{\glsentrytext}[1]{\@gls@entry@field{#1}{text}}

```

`\Glsentrytext`

```

4323 \newrobustcmd*{\Glsentrytext}[1]{%
4324   \@Gls@entry@field{#1}{text}%
4325 }

```

Get the plural form:

\glentryplural

```
4326 \newcommand*{\glentryplural}[1]{%
4327   \@gls@entry@field{#1}{plural}%
4328 }
```

\Glsentryplural

```
4329 \newrobustcmd*{\Glsentryplural}[1]{%
4330   \@Gls@entry@field{#1}{plural}%
4331 }
```

Get the symbol associated with this entry. The argument is the label associated with the entry.

\glentrysymbol

```
4332 \newcommand*{\glentrysymbol}[1]{%
4333   \@gls@entry@field{#1}{symbol}%
4334 }
```

\Glsentrysymbol

```
4335 \newrobustcmd*{\Glsentrysymbol}[1]{%
4336   \@Gls@entry@field{#1}{symbol}%
4337 }
```

Plural form:

trysymbolplural

```
4338 \newcommand*{\glentrysymbolplural}[1]{%
4339   \@gls@entry@field{#1}{symbolplural}%
4340 }
```

trysymbolplural

```
4341 \newrobustcmd*{\Glsentrysymbolplural}[1]{%
4342   \@Gls@entry@field{#1}{symbolplural}%
4343 }
```

Get the entry text to be used when the entry is first used in the document (as specified by the first key when the entry was defined).

\glentryfirst

```
4344 \newcommand*{\glentryfirst}[1]{%
4345   \@gls@entry@field{#1}{first}%
4346 }
```

\Glsentryfirst

```
4347 \newrobustcmd*{\Glsentryfirst}[1]{%
4348   \@Gls@entry@field{#1}{first}%
4349 }
```

Get the plural form (as specified by the firstplural key when the entry was defined).

entryfirstplural

```
4350 \newcommand*{\glentryfirstplural}[1]{%
4351   \@gls@entry@field{#1}{firstpl}%
4352 }
```

entryfirstplural

```
4353 \newrobustcmd*{\Glsentryfirstplural}[1]{%
4354   \@Gls@entry@field{#1}{firstpl}%
4355 }
```

entrytitlecase

```
4356 \newrobustcmd*{\@glentrytitlecase}[2]{%
4357   \glsfieldfetch{#1}{#2}{\@gls@value}%
4358   \xcapitalisewords{\@gls@value}%
4359 }
4360 \ifdef\texorpdfstring
4361 {
4362   \newcommand*{\glentrytitlecase}[2]{%
4363     \texorpdfstring
4364       {\@glentrytitlecase{#1}{#2}}%
4365       {\@gls@entry@field{#1}{#2}}%
4366   }
4367 }
4368 {
4369   \newcommand*{\glentrytitlecase}[2]{\@glentrytitlecase{#1}{#2}}
4370 }
```

Display the glossary type with which this entry is associated (as specified by the type key used when the entry was defined)

\glentrytype

```
4371 \newcommand*{\glentrytype}[1]{\@gls@entry@field{#1}{type}}
```

Display the sort text used for this entry. Note that the sort key is sanitize, so unexpected results may occur if the sort key contained commands.

\glentrysort

```
4372 \newcommand*{\glentrysort}[1]{%
4373   \@gls@entry@field{#1}{sort}%
4374 }
```

\glentryuseri Get the first user key (as specified by the user1 when the entry was defined). The argument is the label associated with the entry.

```
4375 \newcommand*{\glentryuseri}[1]{%
4376   \@gls@entry@field{#1}{useri}%
4377 }
```

\Glsentryuseri

```
4378 \newrobustcmd*{\Glsentryuseri}[1]{%
```

```

4379 \@Gls@entry@field{#1}{useri}%
4380 }

```

`\glentryuserii` Get the second user key (as specified by the user2 when the entry was defined). The argument is the label associated with the entry.

```

4381 \newcommand*{\glentryuserii}[1]{%
4382 \@Gls@entry@field{#1}{userii}%
4383 }

```

`\Glsentryuserii`

```

4384 \newrobustcmd*{\Glsentryuserii}[1]{%
4385 \@Gls@entry@field{#1}{userii}%
4386 }

```

`glentryuseriii` Get the third user key (as specified by the user3 when the entry was defined). The argument is the label associated with the entry.

```

4387 \newcommand*{\glentryuseriii}[1]{%
4388 \@Gls@entry@field{#1}{useriii}%
4389 }

```

`Glsentryuseriii`

```

4390 \newrobustcmd*{\Glsentryuseriii}[1]{%
4391 \@Gls@entry@field{#1}{useriii}%
4392 }

```

`\glentryuseriv` Get the fourth user key (as specified by the user4 when the entry was defined). The argument is the label associated with the entry.

```

4393 \newcommand*{\glentryuseriv}[1]{%
4394 \@Gls@entry@field{#1}{useriv}%
4395 }

```

`\Glsentryuseriv`

```

4396 \newrobustcmd*{\Glsentryuseriv}[1]{%
4397 \@Gls@entry@field{#1}{useriv}%
4398 }

```

`\glentryuserv` Get the fifth user key (as specified by the user5 when the entry was defined). The argument is the label associated with the entry.

```

4399 \newcommand*{\glentryuserv}[1]{%
4400 \@Gls@entry@field{#1}{userv}%
4401 }

```

`\Glsentryuserv`

```

4402 \newrobustcmd*{\Glsentryuserv}[1]{%
4403 \@Gls@entry@field{#1}{userv}%
4404 }

```

`\glentryuservi` Get the sixth user key (as specified by the user6 when the entry was defined). The argument is the label associated with the entry.

```
4405 \newcommand*{\glentryuservi}[1]{%
4406   \@gls@entry@field{#1}{uservi}%
4407 }
```

`\Glsentryuservi`

```
4408 \newrobustcmd*{\Glsentryuservi}[1]{%
4409   \@Gls@entry@field{#1}{uservi}%
4410 }
```

`\glentryshort` Get the short key (as specified by the short the entry was defined). The argument is the label associated with the entry.

```
4411 \newcommand*{\glentryshort}[1]{\@gls@entry@field{#1}{short}}
```

`\Glsentryshort`

```
4412 \newrobustcmd*{\Glsentryshort}[1]{%
4413   \@Gls@entry@field{#1}{short}%
4414 }
```

`glentryshortpl` Get the short plural key (as specified by the shortplural the entry was defined). The argument is the label associated with the entry.

```
4415 \newcommand*{\glentryshortpl}[1]{\@gls@entry@field{#1}{shortpl}}
```

`Glsentryshortpl`

```
4416 \newrobustcmd*{\Glsentryshortpl}[1]{%
4417   \@Gls@entry@field{#1}{shortpl}%
4418 }
```

`\glentrylong` Get the long key (as specified by the long the entry was defined). The argument is the label associated with the entry.

```
4419 \newcommand*{\glentrylong}[1]{\@gls@entry@field{#1}{long}}
```

`\Glsentrylong`

```
4420 \newrobustcmd*{\Glsentrylong}[1]{%
4421   \@Gls@entry@field{#1}{long}%
4422 }
```

`\glentrylongpl` Get the long plural key (as specified by the longplural the entry was defined). The argument is the label associated with the entry.

```
4423 \newcommand*{\glentrylongpl}[1]{\@gls@entry@field{#1}{longpl}}
```

`\Glsentrylongpl`

```
4424 \newrobustcmd*{\Glsentrylongpl}[1]{%
4425   \@Gls@entry@field{#1}{longpl}%
4426 }
```

Short cut macros to access full form:

`\glsentryfull`

```
4427 \newcommand*{\glsentryfull}[1]{%
4428   \acrfullformat{\glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4429 }
```

`\Glsentryfull`

```
4430 \newrobustcmd*{\Glsentryfull}[1]{%
4431   \acrfullformat{\Glsentrylong{#1}}{\acronymfont{\glsentryshort{#1}}}%
4432 }
```

`\glsentryfullpl`

```
4433 \newcommand*{\glsentryfullpl}[1]{%
4434   \acrfullformat{\glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4435 }
```

`\Glsentryfullpl`

```
4436 \newrobustcmd*{\Glsentryfullpl}[1]{%
4437   \acrfullformat{\Glsentrylongpl{#1}}{\acronymfont{\glsentryshortpl{#1}}}%
4438 }
```

`entrynumberlist` Displays the number list as is.

```
4439 \newcommand*{\glsentrynumberlist}[1]{%
4440   \glsdoifexists{#1}%
4441   {%
4442     \@gls@entry@field{#1}{numberlist}%
4443   }%
4444 }
```

`splaynumberlist` Formats the number list for the given entry label. Doesn't work with hyperref.

```
4445 \@ifpackageloaded{hyperref} {%
4446   \newcommand*{\glsdisplaynumberlist}[1]{%
4447     \GlossariesWarning
4448     {%
4449       \string\glsdisplaynumberlist\space
4450       doesn't work with hyperref.^^JUsing
4451       \string\glsentrynumberlist\space instead%
4452     }%
4453     \glsentrynumberlist{#1}%
4454   }%
4455 }%
4456 {%
4457   \newcommand*{\glsdisplaynumberlist}[1]{%
4458     \glsdoifexists{#1}%
4459     {%
4460       \bgroup
```



```

4496 \define@key{glossadd}{counter}{\def\@gls@counter{#1}}
4497 \define@key{glossadd}{format}{\def\@glsnumberformat{#1}}

```

This key is only used by `\glsaddall`:

```

4498 \define@key{glossadd}{types}{\def\@glo@type{#1}}

```

`\glsadd[<options>]{<label>}`

Add a term to the glossary without generating any link text. The optional argument indicates which counter to use, and how to format it (using a key-value list) the second argument is the entry label. Note that *<options>* only has two keys: counter and format (the types key will be ignored).

`\glsadd`

```

4499 \newrobustcmd*{\glsadd}[2] [] {%

```

Need to move to horizontal mode if not already in it, but only if not in preamble.

```

4500 \@gls@adjustmode
4501 \glsdoifexists{#2}%
4502 {%
4503 \def\@glsnumberformat{glsnumberformat}%
4504 \edef\@gls@counter{\csname glo@%glsdetoklabel{#2}@counter\endcsname}%
4505 \setkeys{glossadd}{#1}%

```

Store the entry's counter in `\theglsentrycounter`

```

4506 \@gls@saveentrycounter

```

This should use `\@do@wrglossary` rather than `\do@wrglossary` since the whole point of `\glsadd` is to add a line to the glossary.

```

4507 \@@do@wrglossary{#2}%
4508 }%
4509 }

```

`@gls@adjustmode`

```

4510 \newcommand*{\@gls@adjustmode}{}
4511 \AtBeginDocument{\renewcommand*{\@gls@adjustmode}{\ifvmode\mbox{}\fi}}

```

`\glsaddall[<option list>]`

Add all terms defined for the listed glossaries (without displaying any text). If types key is omitted, apply to all glossary types.

`\glsaddall`

```

4512 \newrobustcmd*{\glsaddall}[1] [] {%
4513 \edef\@glo@type{\@glo@types}%
4514 \setkeys{glossadd}{#1}%
4515 \forallglsentries[\@glo@type]{\@glo@entry}{%

```

```

4516 \glsadd[#1]{\@glo@entry}%
4517 }%
4518 }

```

```

\glsaddallunused \glsaddallunused[<glossary type>]

```

Add all used terms defined for the listed glossaries (without displaying any text). If optional argument is omitted, apply to all glossary types. This should typically go at the end of the document.

```

4519 \newrobustcmd*{\glsaddallunused}[1][\@glo@types]{%
4520 \forallglsentries[#1]{\@glo@entry}%
4521 {%
4522 \ifglsused{\@glo@entry}{\glsadd[format=glsignore]{\@glo@entry}}%
4523 }%
4524 }

```

```

\glsignore

```

```

4525 \newcommand*{\glsignore}[1]{}

```

1.13 Creating associated files

The `\writeist` command creates the associated customized `.ist` makeindex style file. While defining this command, some characters have their catcodes temporarily changed to ensure they get written to the `.ist` file correctly. The makeindex actual character (usually `@`) is redefined to be a `?`, to allow internal commands to be written to the glossary file output file.

The special characters are stored in `\@gls@actualchar`, `\@gls@encapchar`, `\@gls@levelchar` and `\@gls@quotechar` to make them easier to use later, but don't change these values, because the characters are encoded in the command definitions that are used to escape the special characters (which means that the user no longer needs to worry about makeindex special characters).

The symbols and numbers label for group headings are hardwired into the `.ist` file as `glssymbols` and `glsnumbers`, the group titles can be translated (so that `\glssymbolsgroupname` replaces `glssymbols` and `\glsnumbersgroupname` replaces `glsnumbers`) using the command `\glsgroupname` which is defined in `.` This is done to prevent any problem characters in `\glssymbolsgroupname` and `\glsnumbersgroupname` from breaking hyperlinks.

```

\glsopenbrace Define \glsopenbrace to make it easier to write an opening brace to a file.

```

```

4526 \edef\glsopenbrace{\expandafter\@gobble\string\{ }

```

```

\glsclosebrace Define \glsclosebrace to make it easier to write an opening brace to a file.

```

```

4527 \edef\glsclosebrace{\expandafter\@gobble\string\} }

```

```

\glsbackslash Define \glsbackslash to make it easier to write a backslash to a file.

```

```

4528 \edef\glsbackslash{\expandafter\@gobble\string\ }

```

`\glsquote` Define command that makes it easier to write quote marks to a file in the event that the double quote character has been made active.

```
4529 \edef\glsquote#1{\string"#1\string"}
```

`\glspercentchar` Define `\glspercentchar` to make it easier to write a percent character to a file.

```
4530 \edef\glspercentchar{\expandafter\@gobble\string\%}
```

`\glstildechar` Define `\glstildechar` to make it easier to write a tilde character to a file.

```
4531 \edef\glstildechar{\string~}
```

`@glsfirstletter` Define the first letter to come after the digits 0,...,9. Only required for xindy.

```
4532 \ifglsxindy
4533   \newcommand*{\@glsfirstletter}{A}
4534 \fi
```

`letterAfterDigits` Sets the first letter to come after the digits 0,...,9.

```
4535 \ifglsxindy
4536   \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
4537     \renewcommand*{\@glsfirstletter}{#1}}
4538 \else
4539   \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
4540     \glsnoxywarning\GlsSetXdyFirstLetterAfterDigits}
4541 \fi
```

`\@glsminrange` Define the minimum number of successive location references to merge into a range.

```
4542 \newcommand*{\@glsminrange}{2}
```

`yMinRangeLength` Set the minimum range length. The value must either be none or a positive integer. The glossaries package doesn't check if the argument is valid, that is left to xindy.

```
4543 \ifglsxindy
4544   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4545     \renewcommand*{\@glsminrange}{#1}}
4546 \else
4547   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
4548     \glsnoxywarning\GlsSetXdyMinRangeLength}
4549 \fi
```

`\writeist`

```
4550 \ifglsxindy
    Code to use if xindy is required.
4551   \def\writeist{%
    Define write register if not already defined
4552     \ifundef{\glswrite}{\newwrite\glswrite}{}%
    Update attributes list
4553     \@gls@addpredefinedattributes
```

Open the file.

```
4554 \openout\glswrite=\istfilename
```

Write header comment at the start of the file

```
4555 \write\glswrite{;; xindy style file created by the glossaries
4556 package}%
4557 \write\glswrite{;; for document '\jobname' on
4558 \the\year-\the\month-\the\day}%
```

Specify the required styles

```
4559 \write\glswrite{^^J; required styles^^J}
4560 \@for\@xdystyle:=\@xdyrequiredstyles\do{%
4561 \ifx\@xdystyle\@empty
4562 \else
4563 \protected@write\glswrite{}{(require
4564 \string"\@xdystyle.xdy\string")}%
4565 \fi
4566 }%
```

List the allowed attributes (possible values used by the format key)

```
4567 \write\glswrite{^^J%
4568 ; list of allowed attributes (number formats)^^J}%
4569 \write\glswrite{(define-attributes ((\@xdyattributes)))}%
```

Define any additional alphabets

```
4570 \write\glswrite{^^J; user defined alphabets^^J}%
4571 \write\glswrite{\@xdyuseralphabets}%
```

Define location classes.

```
4572 \write\glswrite{^^J; location class definitions^^J}%
```

As from version 3.0, locations are now specified as $\{\langle Hprefix \rangle\}\{\langle number \rangle\}$, so need to add all possible combinations of location types.

```
4573 \@for\@gls@classI:=\@gls@xdy@locationlist\do{%
```

Case where $\langle Hprefix \rangle$ is empty:

```
4574 \protected@write\glswrite{}{(define-location-class
4575 \string"\@gls@classI\string"^^J\space\space\space
4576 (
4577 :sep "{}{"
4578 \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4579 :sep "}"
4580 )
4581 ^^J\space\space\space
4582 :min-range-length \@glsminrange^^J%
4583 )
4584 }%
```

Nested iteration over all classes:

```
4585 {%
4586 \@for\@gls@classII:=\@gls@xdy@locationlist\do{%
4587 \protected@write\glswrite{}{(define-location-class
```

```

4588         \string"\@gls@classII-\@gls@classI\string"
4589         ^^J\space\space\space
4590     (
4591         :sep "{"
4592         \csname @gls@xdy@Lclass@\@gls@classII\endcsname\space
4593         :sep "{"
4594         \csname @gls@xdy@Lclass@\@gls@classI\endcsname\space
4595         :sep "}"
4596     )
4597     ^^J\space\space\space
4598     :min-range-length \@glsminrange^^J%
4599 )
4600 }%
4601 }%
4602 }%
4603 }%

```

User defined location classes (needs checking for new location format).

```

4604 \write\glswrite{^^J; user defined location classes}%
4605 \write\glswrite{\@xdyuserlocationdefs}%

```

Cross-reference class. (The unverified option is used as the cross-references are supplied using the list of labels along with the optional argument for `\glsseeformat` which xindy won't recognise.)

```

4606 \write\glswrite{^^J; define cross-reference class^^J}%
4607 \write\glswrite{(define-crossref-class \string"see\string"
4608     :unverified )}%

```

Define how cross-references should be displayed. This adds an empty set of braces after the cross-referencing information allowing for the final argument of `\glsseeformat` which gets ignored. (When using `makeindex` this final argument contains the location information which is not required.)

```

4609 \write\glswrite{(markup-crossref-list
4610     :class \string"see\string"^^J\space\space\space
4611     :open \string"\string\glsseeformat\string"
4612     :close \string"{}\string")}%

```

List the order to sort the classes.

```

4613 \write\glswrite{^^J; define the order of the location classes}%
4614 \write\glswrite{(define-location-class-order
4615     (\@xdylocationclassorder))}%

```

Specify what to write to the start and end of the glossary file.

```

4616 \write\glswrite{^^J; define the glossary markup^^J}%
4617 \write\glswrite{(markup-index^^J\space\space\space
4618     :open \string"\string
4619     \glossarysection[\string\glossarytoctitle]{\string
4620     \glossarytitle}\string\glossarypreamble}%

```

Add all the xindy-only macro definitions (needed to prevent errors in the event that the user changes from xindy to makeindex)

```

4621 \@for\@this@ctr:=\@xdycounters\do{%
4622   {%
4623     \@for\@this@attr:=\@xdyattributelist\do{%
4624       \protected@write\glswrite{}\string\providecommand*%
4625         \expandafter\string
4626         \csname glsX\@this@ctr X\@this@attr\endcsname[2]%
4627         {%
4628           \string\setentrycounter
4629             [\expandafter@gobble\string\#1]{\@this@ctr}%
4630           \expandafter\string
4631           \csname\@this@attr\endcsname
4632             {\expandafter@gobble\string\#2}%
4633         }%
4634     }%
4635 }%
4636 }%
4637 }%

```

Add the end part of the open tag and the rest of the markup-index information:

```

4638 \write\glswrite{%
4639   \string\begin
4640     {theglossary}\string\glossaryheader\glstildechar n\string" ^^J\space
4641     \space\space:close \string"\glspercentchar\glstildechar n\string
4642     \end{theglossary}\string\glossarypostamble
4643     \glstildechar n\string" ^^J\space\space\space
4644     :tree)}}%

```

Specify what to put between letter groups

```

4645 \write\glswrite{(markup-letter-group-list
4646   :sep \string"\string\glsgroupskip\glstildechar n\string")}%

```

Specify what to put between entries

```

4647 \write\glswrite{(markup-indexentry
4648   :open \string"\string\relax \string\glsresetentrylist
4649   \glstildechar n\string")}%

```

Specify how to format entries

```

4650 \write\glswrite{(markup-locclass-list :open
4651   \string"\glsopenbrace\string\glossaryentrynumbers
4652   \glsopenbrace\string\relax\space \string"^^J\space\space\space
4653   :sep \string", \string"
4654   :close \string"\glsclosebrace\glsclosebrace\string")}%

```

Specify how to separate location numbers

```

4655 \write\glswrite{(markup-locref-list
4656   :sep \string"\string\delimN\space\string")}%

```

Specify how to indicate location ranges

```

4657 \write\glswrite{(markup-range
4658   :sep \string"\string\delimR\space\string")}%

```

Specify 2-page and 3-page suffixes, if defined. First, the values must be sanitized to write them explicitly.

```

4659 \@onelevel@sanitize\gls@suffixF
4660 \@onelevel@sanitize\gls@suffixFF

4661 \ifx\gls@suffixF\@empty
4662 \else
4663 \write\glswrite{(markup-range
4664 :close "\gls@suffixF" :length 1 :ignore-end)}}%
4665 \fi
4666 \ifx\gls@suffixFF\@empty
4667 \else
4668 \write\glswrite{(markup-range
4669 :close "\gls@suffixFF" :length 2 :ignore-end)}}%
4670 \fi

```

Specify how to format locations.

```

4671 \write\glswrite{^^J; define format to use for locations^^J}%
4672 \write\glswrite{\@xdylocref}%

```

Specify how to separate letter groups.

```

4673 \write\glswrite{^^J; define letter group list format^^J}%
4674 \write\glswrite{(markup-letter-group-list
4675 :sep \string"\string\glsgroupskip\glstildechar n\string")}%

```

Define letter group headings.

```

4676 \write\glswrite{^^J; letter group headings^^J}%
4677 \write\glswrite{(markup-letter-group
4678 :open-head \string"\string\glsgroupheading
4679 \glsopenbrace\string"^^J\space\space\space
4680 :close-head \string"\glsclosebrace\string")}%

```

Define additional letter groups.

```

4681 \write\glswrite{^^J; additional letter groups^^J}%
4682 \write\glswrite{\@xdylettergroups}%

```

Define additional sort rules

```

4683 \write\glswrite{^^J; additional sort rules^^J}
4684 \write\glswrite{\@xdysortrules}%

```

Hook for any additional information:

```

4685 \@gls@writeisthook

```

Close the style file

```

4686 \closeout\glswrite

```

Suppress any further calls.

```

4687 \let\writeist\relax
4688 }
4689 \else

```

Code to use if makeindex is required.

```

4690 \edef\@gls@actualchar{\string?}
4691 \edef\@gls@encapchar{\string|}
4692 \edef\@gls@levelchar{\string!}
4693 \edef\@gls@quotechar{\string"}%
4694 \let\GlsSetQuote\gls@nosetquote
4695 \def\writeist{\relax
4696   \ifundef{\glswrite}{\newwrite\glswrite}{}\relax
4697   \openout\glswrite=\istfilename
4698   \write\glswrite{\glspersentchar\space makeindex style file
4699     created by the glossaries package}
4700   \write\glswrite{\glspersentchar\space for document
4701     '\jobname' on \the\year-\the\month-\the\day}
4702   \write\glswrite{actual '\@gls@actualchar'}
4703   \write\glswrite{encap '\@gls@encapchar'}
4704   \write\glswrite{level '\@gls@levelchar'}
4705   \write\glswrite{quote '\@gls@quotechar'}
4706   \write\glswrite{keyword \string\string\glossaryentry\string}
4707   \write\glswrite{preamble \string\string\glossarysection[\string
4708     \glossarytoctitle]{\string\glossarytitle}\string
4709     \glossarypreamble\string\n\string\begin{theglossary}\string
4710     \glossaryheader\string\n\string}
4711   \write\glswrite{postamble \string\string%\string\n\string
4712     \end{theglossary}\string\glossarypostamble\string\n
4713     \string}
4714   \write\glswrite{group_skip \string\string\glsgroupskip\string\n
4715     \string}
4716   \write\glswrite{item_0 \string\string%\string\n\string}
4717   \write\glswrite{item_1 \string\string%\string\n\string}
4718   \write\glswrite{item_2 \string\string%\string\n\string}
4719   \write\glswrite{item_01 \string\string%\string\n\string}
4720   \write\glswrite{item_x1
4721     \string\string\relax \string\glsresetentrylist\string\n
4722     \string}
4723   \write\glswrite{item_12 \string\string%\string\n\string}
4724   \write\glswrite{item_x2
4725     \string\string\relax \string\glsresetentrylist\string\n
4726     \string}

4727   \write\glswrite{delim_0 \string\string\{\string
4728     \glossaryentrynumbers\string\{\string\relax \string}
4729   \write\glswrite{delim_1 \string\string\{\string
4730     \glossaryentrynumbers\string\{\string\relax \string}
4731   \write\glswrite{delim_2 \string\string\{\string
4732     \glossaryentrynumbers\string\{\string\relax \string}
4733   \write\glswrite{delim_t \string\string\}\string\}\string}
4734   \write\glswrite{delim_n \string\string\delimN \string}
4735   \write\glswrite{delim_r \string\string\delimR \string}
4736   \write\glswrite{headings_flag 1}
4737   \write\glswrite{heading_prefix

```



```

4738     \string"\string\glsgroupheading\string\{\string"}
4739 \write\glswrite{heading_suffix
4740     \string"\string}\string\relax
4741     \string\glsgroupheading\string"}
4742 \write\glswrite{symhead_positive \string"glssymbols\string"}
4743 \write\glswrite{numhead_positive \string"glslnumbers\string"}
4744 \write\glswrite{page_compositor \string"glscpositor\string"}
4745 \@gls@escbsdq\gls@suffixF
4746 \@gls@escbsdq\gls@suffixFF
4747 \ifx\gls@suffixF\@empty
4748 \else
4749     \write\glswrite{suffix_2p \string"\gls@suffixF\string"}
4750 \fi
4751 \ifx\gls@suffixFF\@empty
4752 \else
4753     \write\glswrite{suffix_3p \string"\gls@suffixFF\string"}
4754 \fi

```

Hook for any additional information:

```

4755 \@gls@writeisthook

```

Close the file and disable \writeist.

```

4756 \closeout\glswrite
4757 \let\writeist\relax
4758 }
4759 \fi

```

SetWriteIstHook Allow user to append information to the style file.

```

4760 \newcommand*\GlsSetWriteIstHook[1]{\renewcommand*\@gls@writeisthook{#1}}
4761 \@onlypremake\GlsSetWriteIstHook

```

ls@writeisthook

```

4762 \newcommand*\@gls@writeisthook{}

```

\GlsSetQuote Allow user to set the makeindex quote character. This is primarily for ngerman users who want to use makeindex's -g option.

```

4763 \ifglxindy
4764 \newcommand*\GlsSetQuote[1]{\glsnomakeindexwarning\GlsSetQuote}
4765 \newcommand*\gls@nosetquote[1]{\glsnomakeindexwarning\GlsSetQuote}
4766 \else
4767 \newcommand*\GlsSetQuote[1]{\edef\@gls@quotechar{\string#1}%

```

If German is in use, set the extra makeindex option so makeglossaries can pick it up.

```

4768 \@ifpackageloaded{tracklang}%
4769 {%
4770 \IfTrackedLanguage{german}%
4771 {%
4772 \def\@gls@extramakeindexopts{-g}%
4773 }%
4774 }%

```

```

4775 }%
4776 {}%

Need to redefine \@gls@checkquote
4777 \edef\@gls@docheckquotedef{%
4778 \noexpand\def\noexpand\@gls@checkquote####1#1####2#1####3\noexpand\null{%
4779 \noexpand\@gls@tmpb=\noexpand\expandafter{\noexpand\@gls@checkedmkidx}%
4780 \noexpand\toks@={####1}%
4781 \noexpand\ifx\noexpand\null####2\noexpand\null
4782 \noexpand\ifx\noexpand\null####3\noexpand\null
4783 \noexpand\edef\noexpand\@gls@checkedmkidx{%
4784 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
4785 \noexpand\def\noexpand\@gls@checkquote{\noexpand\relax}%
4786 \noexpand\else
4787 \noexpand\edef\noexpand\@gls@checkedmkidx{%
4788 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4789 \noexpand\@gls@quotechar\noexpand\@gls@quotechar
4790 \noexpand\@gls@quotechar\noexpand\@gls@quotechar}%
4791 \noexpand\def\noexpand\@gls@checkquote{%
4792 \noexpand\@gls@checkquote####3\noexpand\null}%
4793 \noexpand\fi
4794 \noexpand\else
4795 \noexpand\edef\noexpand\@gls@checkedmkidx{%
4796 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4797 \noexpand\@gls@quotechar\noexpand\@gls@quotechar}%
4798 \noexpand\ifx\noexpand\null####3\noexpand\null
4799 \noexpand\def\noexpand\@gls@checkquote{%
4800 \noexpand\@gls@checkquote####2#1#1\noexpand\null}%
4801 \noexpand\else
4802 \noexpand\def\noexpand\@gls@checkquote{%
4803 \noexpand\@gls@checkquote####2#1####3\noexpand\null}%
4804 \noexpand\fi
4805 \noexpand\fi
4806 \noexpand\@gls@checkquote
4807 }%
4808 }%
4809 \@gls@docheckquotedef
4810 \edef\@gls@docheckquotedef{%
4811 \noexpand\renewcommand{\noexpand\@gls@checkmkidxchars}[1]{%
4812 \noexpand\def\noexpand\@gls@checkedmkidx{%
4813 \noexpand\expandafter\noexpand\@gls@checkquote####1\noexpand\@nil
4814 #1#1\noexpand\null
4815 \noexpand\expandafter\noexpand\@gls@updatechecked
4816 \noexpand\@gls@checkedmkidx{####1}%
4817 \noexpand\def\noexpand\@gls@checkedmkidx{%
4818 \noexpand\expandafter\noexpand\@gls@checkescquote####1\noexpand\@nil
4819 \expandonce{\csname#1\endcsname}\expandonce{\csname#1\endcsname}%
4820 \noexpand\null
4821 \noexpand\expandafter\noexpand\@gls@updatechecked
4822 \noexpand\@gls@checkedmkidx{####1}%

```

```

4823 \noexpand\def\noexpand\@gls@checkedmkidx{%
4824 \noexpand\expandafter\noexpand\@gls@checkescactual####1\noexpand\@nil
4825 \noexpand\?\noexpand\?\noexpand\null
4826 \noexpand\expandafter\noexpand\@gls@updatechecked
4827 \noexpand\@gls@checkedmkidx{####1}%
4828 \noexpand\def\noexpand\@gls@checkedmkidx{%
4829 \noexpand\expandafter\noexpand\@gls@checkactual####1\noexpand\@nil
4830 \noexpand?\noexpand?\noexpand\null
4831 \noexpand\expandafter\noexpand\@gls@updatechecked
4832 \noexpand\@gls@checkedmkidx{####1}%
4833 \noexpand\def\noexpand\@gls@checkedmkidx{%
4834 \noexpand\expandafter\noexpand\@gls@checkbar####1\noexpand\@nil
4835 \noexpand|\noexpand|\noexpand\null
4836 \noexpand\expandafter\noexpand\@gls@updatechecked
4837 \noexpand\@gls@checkedmkidx{####1}%
4838 \noexpand\def\noexpand\@gls@checkedmkidx{%
4839 \noexpand\expandafter\noexpand\@gls@checkescbar####1\noexpand\@nil
4840 \noexpand||\noexpand||\noexpand\null
4841 \noexpand\expandafter\noexpand\@gls@updatechecked
4842 \noexpand\@gls@checkedmkidx{####1}%
4843 \noexpand\def\noexpand\@gls@checkedmkidx{%
4844 \noexpand\expandafter\noexpand\@gls@checklevel####1\noexpand\@nil
4845 \noexpand!\noexpand!\noexpand\null
4846 \noexpand\expandafter\noexpand\@gls@updatechecked
4847 \noexpand\@gls@checkedmkidx{####1}%
4848 }%
4849 }%
4850 \@gls@docheckquotedef
4851 \edef\@gls@docheckquotedef{%
4852 \noexpand\def\noexpand\@gls@checkescquote####1%
4853 \expandonce{\csname#1\endcsname}####2\expandonce{\csname#1\endcsname}%
4854 ####3\noexpand\null{%
4855 \noexpand\@gls@tmpb=\noexpand\expandafter{\noexpand\@gls@checkedmkidx}%
4856 \noexpand\toks@={####1}%
4857 \noexpand\ifx\noexpand\null####2\noexpand\null
4858 \noexpand\ifx\noexpand\null####3\noexpand\null
4859 \noexpand\edef\noexpand\@gls@checkedmkidx{%
4860 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
4861 \noexpand\def\noexpand\@gls@checkescquote{\noexpand\relax}%
4862 \noexpand\else
4863 \noexpand\edef\noexpand\@gls@checkedmkidx{%
4864 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4865 \noexpand\@gls@quotechar\noexpand\string\expandonce{%
4866 \csname#1\endcsname}\noexpand\@gls@quotechar
4867 \noexpand\@gls@quotechar\noexpand\string\expandonce{%
4868 \csname#1\endcsname}\noexpand\@gls@quotechar}%
4869 \noexpand\def\noexpand\@gls@checkescquote{%
4870 \noexpand\@gls@checkescquote####3\noexpand\null}%
4871 \noexpand\fi

```

```

4872 \noexpand\else
4873 \noexpand\edef\noexpand\@gls@checkedmkidx{%
4874 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
4875 \noexpand\@gls@quotearch\noexpand\string
4876 \expandonce{\csname#1\endcsname}\noexpand\@gls@quotearch}%
4877 \noexpand\ifx\noexpand\null####3\noexpand\null
4878 \noexpand\def\noexpand\@gls@checkescquote{%
4879 \noexpand\@gls@checkescquote####2\expandonce{\csname#1\endcsname}%
4880 \expandonce{\csname#1\endcsname}\noexpand\null}%
4881 \noexpand\else
4882 \noexpand\def\noexpand\@gls@checkescquote{%
4883 \noexpand\@gls@checkescquote####2\expandonce{\csname#1\endcsname}%
4884 ####3\noexpand\null}%
4885 \noexpand\fi
4886 \noexpand\fi
4887 \noexpand\@gls@checkescquote
4888 }%
4889 }%
4890 \@gls@docheckquotedef
4891 }
4892 \newcommand*{\gls@nosetquote}[1]{\PackageError{glossaries}%
4893 {\string\GlsSetQuote\space not permitted here}%
4894 {Move \string\GlsSetQuote\space earlier in the preamble, as
4895 soon as possible after glossaries.sty has been loaded}}
4896 \fi

```

ramakeindexopts

```

4897 \newcommand*{\@gls@extramakeindexopts}[1]{}

```

The command `\noist` will suppress the creation of the `.ist` file. Obviously you need to use this command before `\writeist` to have any effect.

`\noist`

```

4898 \newcommand{\noist}{%
  Update attributes list
4899 \@gls@addpredefinedattributes
4900 \let\writeist\relax
4901 }

```

`\@makeglossary` is an internal command that takes an argument indicating the glossary type. This command will create the glossary file required by `makeindex` for the given glossary type, using the extension supplied by the `<out-ext>` parameter used in `\newglossary` (and it will also activate the `\glossary` command, and create the customized `.ist` `makeindex` style file).

Note that you can't use `\@makeglossary` for only some of the defined glossaries. You either need to have a `\makeglossary` for all glossaries or none (otherwise you will end up with a situation where \TeX is trying to write to a non-existent file). The relevant glossary must be defined prior to using `\@makeglossary`.

\@makeglossary

```
4902 \newcommand*{\@makeglossary}[1]{%
4903   \ifglossaryexists{#1}%
4904   {%
```

Only create a new write if savewrites=false otherwise create a token to collect the information.

```
4905   \ifglssavewrites
4906     \expandafter\newtoks\csname glo@#1@filetok\endcsname
4907   \else
4908     \expandafter\newwrite\csname glo@#1@file\endcsname
4909     \expandafter\@glsopenfile\csname glo@#1@file\endcsname{#1}%
4910   \fi
4911   \@gls@renewglossary
4912   \writeist
4913 }%
4914 {%
4915   \PackageError{glossaries}%
4916   {Glossary type ‘#1’ not defined}%
4917   {New glossaries must be defined before using \string\makeglossary}%
4918 }%
4919 }
```

\@glsopenfile Open write file associated with the given glossary.

```
4920 \newcommand*{\@glsopenfile}[2]{%
4921   \immediate\openout#1=\jobname.\csname @glotype@#2@out\endcsname
4922   \PackageInfo{glossaries}{Writing glossary file
4923     \jobname.\csname @glotype@#2@out\endcsname}%
4924 }
```

\@closegls

```
4925 \newcommand*{\@closegls}[1]{%
4926   \closeout\csname glo@#1@file\endcsname
4927 }
```

\@gls@automake

```
4928 \ifglxindy
4929 \newcommand*{\@gls@automake}[1]{%
4930   \ifglossaryexists{#1}
4931   {%
4932     \@closegls{#1}%
4933     \ifdefstring{\glsorder}{letter}%
4934     {\def\@gls@order{-M ord/letorder }}%
4935     {\let\@gls@order\@empty}%
4936     \ifcsundef{@xdy@#1@language}%
4937     {\let\@gls@langmod\@xdy@main@language}%
4938     {\letcs\@gls@langmod{@xdy@#1@language}}%
4939     \edef\@gls@dothiswrite{\noexpand\write18{xindy
4940       -I xindy
```

```

4941      \@gls@order
4942      -L \@gls@langmod\space
4943      -M \@gls@istfilebase\space
4944      -C \@gls@codepage\space
4945      -t \jobname.\csuse{@glotype@#1@log}
4946      -o \jobname.\csuse{@glotype@#1@in}
4947      \jobname.\csuse{@glotype@#1@out}}}%
4948  }%
4949  \@gls@dothiswrite
4950 }%
4951 {%
4952   \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
4953 }%
4954 }
4955 \else
4956 \newcommand*{\@gls@automake}[1]{%
4957   \ifglossaryexists{#1}
4958   {%
4959     \@closegls{#1}%
4960     \ifdefstring{\glsorder}{letter}%
4961     {\def\@gls@order{-l }}%
4962     {\let\@gls@order\@empty}%
4963     \edef\@gls@dothiswrite{\noexpand\write18{makeindex \@gls@order
4964       -s \istfilename\space
4965       -t \jobname.\csuse{@glotype@#1@log}
4966       -o \jobname.\csuse{@glotype@#1@in}
4967       \jobname.\csuse{@glotype@#1@out}}}%
4968   }%
4969   \@gls@dothiswrite
4970 }%
4971 {%
4972   \GlossariesWarning{Can't make glossary '#1', it doesn't exist}%
4973 }%
4974 }
4975 \fi

```

`\makeglossaries` Issue warning that `\makeglossaries` hasn't been used.

```

4976 \newcommand*{\@warn@nomakeglossaries}{}

```

Only use this if warning if `\printglossary` has been used without `\makeglossaries`

```

4977 \newcommand*{\@warn@nomakeglossaries}{\@warn@nomakeglossaries}

```

`\makeglossaries` will use `\@makeglossary` for each glossary type that has been defined.

New glossaries need to be defined before using `\makeglossary`, so have `\makeglossaries` redefine `\newglossary` to prevent it being used afterwards.

`\makeglossaries`

```

4978 \newcommand*{\makeglossaries}{%

```

Define the write used for style file also used for all other output files if `savewrites=true`.

```
4979 \ifundef{\glswrite}{\newwrite\glswrite}{}%
```

If the user removes the glossary package from their document, ensure the next run doesn't throw a load of undefined control sequence errors when the aux file is parsed.

```
4980 \protected@write\@auxout{}{\string\providecommand\string\@glsorder[1]{}%
4981 \protected@write\@auxout{}{\string\providecommand\string\@istfilename[1]{}%
```

If \@@gls@extramakeindexopts has been defined, write it:

```
4982 \ifundef\@@gls@extramakeindexopts
4983 {}%
4984 {%
4985 \protected@write\@auxout{}{\string\providecommand
4986 \string\@gls@extramakeindexopts[1]{}%
4987 \protected@write\@auxout{}{\string\@gls@extramakeindexopts
4988 {\@gls@extramakeindexopts}}%
4989 }%
```

Write the name of the style file to the aux file (needed by makeglossaries)

```
4990 \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%
4991 \protected@write\@auxout{}{\string\@glsorder{\glsorder}}%
```

Iterate through each glossary type and activate it.

```
4992 \@for\@glo@type:=\@glo@types\do{%
4993 \ifthenelse{equal{\@glo@type}{}}{}}{%
4994 \@makeglossary{\@glo@type}}%
4995 }%
```

New glossaries must be created before \makeglossaries so disable \newglossary.

```
4996 \renewcommand*\newglossary[4][]{%
4997 \PackageError{glossaries}{New glossaries
4998 must be created before \makeglossaries}{You need
4999 to move \string\makeglossaries\space after all your
5000 \string\newglossary\space commands}}%
```

Any subsequent instances of this command should have no effect

```
5001 \let\@makeglossary\relax
5002 \let\makeglossary\relax
5003 \let\makeglossaries\relax
```

Disable all commands that have no effect after \makeglossaries

```
5004 \@disable@onlypremakeg
```

Allow see key:

```
5005 \let\gls@checkseeallowed\relax
```

Suppress warning about no \makeglossaries

```
5006 \let\warn@nomakeglossaries\relax
```

Activate warning about missing \printglossary

```
5007 \def\warn@noprntglossary{%
5008 \ifdefstring{\@glo@types}{,}%
5009 {%
5010 \GlossariesWarningNoLine{No glossaries have been defined}}%
```

```

5011 }%
5012 {%
5013     \GlossariesWarningNoLine{No \string\printglossary\space
5014     or \string\printglossaries\space
5015     found. ^^J(Remove \string\makeglossaries\space if you
5016     don't want any glossaries.) ^^JThis document will not
5017     have a glossary}%
5018 }%
5019 }%

```

Declare list parser for \glsdisplaynumberlist

```

5020 \ifglssavenumberlist
5021     \edef\@gls@dodolistparser{\noexpand\DeclareListParser
5022     {\noexpand\glsnumlistparser}{\delimN}}}%
5023 \@gls@dodolistparser
5024 \fi

```

Prevent user from also using \makenoidxglossaries

```

5025 \let\makenoidxglossaries\@no@makeglossaries

```

Prohibit sort key in printgloss family:

```

5026 \renewcommand*{\@printgloss@setsort}{%
5027     \let\@glo@assign@sortkey\@glo@no@assign@sortkey
5028 }%

```

Check the automake setting:

```

5029 \ifglsautomake
5030     \renewcommand*{\@gls@doautomake}{%
5031         \@for\@gls@type:=\@glo@types\do{%
5032             \ifdefempty{\@gls@type}{}%
5033             {\@gls@automake{\@gls@type}}}%
5034         }%
5035     }%
5036 \fi
5037 }

```

Must occur in the preamble:

```

5038 \@onlypreamble{\makeglossaries}

```

`\glswrite` The definition of `\glswrite` has now been moved to `\makeglossaries` so that it's only defined if needed.

The `\makeglossary` command is redefined to be identical to `\makeglossaries`. (This is done to reinforce the message that you must either use `\@makeglossary` for all the glossaries or for none of them.)

`\makeglossary`

```

5039 \let\makeglossary\makeglossaries

```

If `\makeglossaries` hasn't been used, issue a warning. Also issue a warning if neither `\printglossaries` nor `\printglossary` have been used.


```

5040 \AtEndDocument{%
5041   \warn@nomakeglossaries
5042   \warn@noprintglossary
5043 }

```

noidxglossaries Analogous to \makeglossaries this activates the commands needed for \printnoidxglossary

```

5044 \newcommand*{\makenoidxglossaries}{%
  Redefine empty glossary warning:
5045   \renewcommand{\@gls@noref@warn}[1]{%
5046     \GlossariesWarning{Empty glossary for
5047       \string\printnoidxglossary[type={##1}].
5048     Rerun may be required (or you may have forgotten to use
5049     commands like \string\gls)}}%
5050 }%

```

Don't escape makeindex/xindy characters

```

5051 \let\@gls@checkmkidxchars\@gobble

```

Write glossary information to aux instead of glossary files

```

5052 \let\@do@wrglossary\gls@noidxglossary

```

Switch on group headings that use the character code:

```

5053 \let\@gls@getgrouptitle\@gls@noidx@getgrouptitle

```

Allow see key:

```

5054 \let\gls@checkseeallowed\relax

```

Redefine cross-referencing macro:

```

5055 \renewcommand{\@do@seeglossary}[2]{%
5056   \edef\@gls@label{\glsdetoklabel{##1}}%
5057   \protected@write\@auxout{}{%
5058     \string\@gls@reference
5059     {\curname glo@\@gls@label @type\endcurname}%
5060     {\@gls@label}%
5061     {%
5062       \string\glsseeformat##2}%
5063     }%
5064   }%
5065 }%

```

If user removes the glossaries package from their document, ensure the next run doesn't throw a load of undefined control sequence errors when the aux file is parsed.

```

5066 \AtBeginDocument
5067 {%
5068   \write\@auxout{\string\providecommand\string\@gls@reference[3]{}}%
5069 }%

```

Change warning about no glossaries

```

5070 \def\warn@noprintglossary{%
5071   \GlossariesWarningNoLine{No \string\printnoidxglossary\space
5072     or \string\printnoidxglossaries ^^J

```

```

5073      found. (Remove \string\makenoidxglossaries\space if you
5074      don't want any glossaries.)^^JThis document will not have a glossary}%
5075 }%

  Suppress warning about no \makeglossaries
5076  \let\warn@nomakeglossaries\relax

  Prevent user from also using \makeglossaries
5077  \let\makeglossaries\@no@makeglossaries

  Allow sort key in printgloss family:
5078  \renewcommand*{\@printgloss@setsort}{%
5079    \let\@glo@assign@sortkey\@glo@assign@sortkey

  Initialise default sort order:
5080    \def\@glo@sorttype{\@glo@default@sorttype}%
5081  }%

  All entries must be defined in the preamble:
5082  \renewcommand*\new@glossaryentry[2]{%
5083    \PackageError{glossaries}{Glossary entries must be
5084      defined in the preamble^^Jwhen you use
5085      \string\makenoidxglossaries}%
5086    {Either move your definitions to the preamble or use
5087      \string\makeglossaries}%
5088  }%

  Redefine \glsentrynumberlist
5089  \renewcommand*{\glsentrynumberlist}[1]{%
5090    \letcs{\@gls@loclist}{glo\@glsdetoklabel{##1}@loclist}%
5091    \ifdef\@gls@loclist
5092    {%
5093      \glsnoidxloclist{\@gls@loclist}%
5094    }%
5095    {%
5096      ??\glsdoifexists{##1}%
5097      {%
5098        \GlossariesWarning{Missing location list for '##1'. Either
5099          a rerun is required or you haven't referenced the entry}%
5100      }%
5101    }%
5102  }%

  Redefine \glsdisplaynumberlist
5103  \renewcommand*{\glsdisplaynumberlist}[1]{%
5104    \letcs{\@gls@loclist}{glo\@glsdetoklabel{##1}@loclist}%
5105    \ifdef\@gls@loclist
5106    {%
5107      \def\@gls@noidxloclist@sep{%
5108        \def\@gls@noidxloclist@sep{%
5109          \def\@gls@noidxloclist@sep{%
5110            \glsnumlistsep

```

```

5111         }%
5112         \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
5113     }%
5114 }%
5115 \def\@gls@noidxloclist@finalsep{}%
5116 \def\@gls@noidxloclist@prev{}%
5117 \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
5118 \@gls@noidxloclist@finalsep
5119 \@gls@noidxloclist@prev
5120 }%
5121 {%
5122 ??\glsdoifexists{##1}%
5123 {%
5124     \GlossariesWarning{Missing location list for ‘##1’. Either
5125         a rerun is required or you haven’t referenced the entry}%
5126 }%
5127 }%
5128 }%

```

Provide a generic way of iterating through the number list:

```

5129 \renewcommand*\@glsnumberlistloop}[3]{%
5130     \letcs{\@gls@loclist}{glo@\glsdetoklabel{##1}@loclist}%
5131     \let\@gls@org\glsnoidxdisplayloc\glsnoidxdisplayloc
5132     \let\@gls@org\glsseeformat\glsseeformat
5133     \let\glsnoidxdisplayloc##2\relax
5134     \let\glsseeformat##3\relax
5135     \ifdef\@gls@loclist
5136     {%
5137         \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
5138     }%
5139     {%
5140         ??\glsdoifexists{##1}%
5141         {%
5142             \GlossariesWarning{Missing location list for ‘##1’. Either
5143                 a rerun is required or you haven’t referenced the entry}%
5144         }%
5145     }%
5146     \let\glsnoidxdisplayloc\@gls@org\glsnoidxdisplayloc
5147     \let\glsseeformat\@gls@org\glsseeformat
5148 }%

```

Modify sanitize sort function

```

5149 \let\@gls@sanitizesort\@gls@noidx@sanitizesort
5150 \let\@gls@nosanitizesort\@gls@noidx@nosanitizesort
5151 \@gls@noidx@setsanitizesort
5152 }

```

Preamble-only command:

```

5153 \@onlypreamble{\makenoidxglossaries}

```

`\lsnumberlistloop` `\glsnumberlistloop{<label>}{<handler>}`

```
5154 \newcommand*{\glsnumberlistloop}[2]{%
5155   \PackageError{glossaries}{\string\glsnumberlistloop\space
5156     only works with \string\makenoidxglossaries}{}%
5157 }
```

`\listloophandler` Handler macro for `\glsnumberlistloop`. (The argument should be in the form `\glsnoidxdisplayloc {<prefix>}{<counter>}{<format>}{<n>}`)

```
5158 \newcommand*{\glsnoidxnumberlistloophandler}[1]{%
5159   #1%
5160 }
```

`@makeglossaries` Can't use both `\makeglossaries` and `\makenoidxglossaries`

```
5161 \newcommand*{\@no@makeglossaries}{%
5162   \PackageError{glossaries}{You can't use both
5163     \string\makeglossaries\space and \string\makenoidxglossaries}%
5164   {Either use one or other (or none) of those commands but not both
5165     together.}%
5166 }
```

`@gls@noref@warn` Warning when no instances of `\@gls@reference` found.

```
5167 \newcommand{\@gls@noref@warn}[1]{%
5168   \GlossariesWarning{\string\makenoidxglossaries\space
5169     is required to make \string\printnoidxglossary[type={#1}] work}%
5170 }
```

`s@noidxglossary` Write the glossary information to the aux file:

```
5171 \newcommand*{\gls@noidxglossary}{%
5172   \protected@write\@auxout{}{%
5173     \string\@gls@reference
5174       {\csname glo@\@gls@label @type\endcsname}%
5175       {\@gls@label}%
5176       {\string\glsnoidxdisplayloc
5177         {\@glo@counterprefix}%
5178         {\@gls@counter}%
5179         {\@glsnumberformat}%
5180         {\@glslocref}%
5181       }%
5182   }%
5183 }
```

1.14 Writing information to associated files

`\istfile` Deprecated.

```
5184 \def\istfile{\glswrite}
```

At the end of the document, the files should be created if savewrites=true.

```
5185 \AtEndDocument{%  
5186   \glswritefiles  
5187 }
```

\@glswritefiles Only write the files if savewrites=true

```
5188 \newcommand*{\@glswritefiles}{%
```

Iterate through all the glossaries

```
5189   \foralllglossaries{\@glo@type}{%
```

Check for empty glossaries (patch provided by Patrick Häcker)

```
5190     \ifcsundef{glo@\@glo@type @filetok}%  
5191     {%  
5192       \def\gls@tmp{}%  
5193     }%  
5194     {%  
5195       \edef\gls@tmp{\expandafter\the  
5196         \csname glo@\@glo@type @filetok\endcsname}%  
5197     }%  
5198     \ifx\gls@tmp\@empty  
5199       \ifx\@glo@type\glsdefaulttype  
5200         \GlossariesWarningNoLine{Glossary '@@glo@type' has no  
5201           entries.^^JRemember to use package option 'nomain' if  
5202 you  
5203           don't want to^^Juse the main glossary}%  
5204       \else  
5205         \GlossariesWarningNoLine{Glossary '@@glo@type' has no  
5206           entries}%  
5207       \fi  
5208     \else  
5209       \@glsopenfile{\glswrite}{\@glo@type}%  
5210       \immediate\write\glswrite{%  
5211         \expandafter\the  
5212         \csname glo@\@glo@type @filetok\endcsname}%  
5213       \immediate\closeout\glswrite  
5214     \fi  
5215   }%  
5216 }
```

As from v4.10, the \glossary command is used by the glossaries package. Since the user isn't expected to use this command (as glossaries takes care of the particular format required for **makeindex/xindy**) there's no need for a user level command. Using a custom internal command prevents any conflict with other packages (and with the \mark mechanism).

In v4.10, the redefinition of \glossary was removed since it wasn't intended as a user level command, however it seems there are packages that have hacked the internal macros used by glossaries and no longer work with this redefinition removed, so it's been restored in v4.11 but is not used at all by glossaries. (This may be removed or moved to a compatibility mode in future.)

`\glossary`

```
5217 \if@gls@docloaded
5218 \else
5219   \renewcommand*{\glossary}[1][main]{\gls@glossary{#1}}
5220 \fi
```

The associated number should be stored in `\theglsentrycounter` before using `\gls@glossary`.

`\gls@glossary`

```
5221 \newcommand*{\gls@glossary}[1]{%
5222   \@gls@glossary{#1}%
5223 }
```

`\@gls@glossary` (In v4.10, `\@glossary` was redefined to `\@gls@glossary` to avoid conflict with other packages.) Define internal `\@gls@glossary` to ignore its argument. This gets redefined in `\@makeglossary`. This is defined to just `\index` as memoir changes the definition of `\@index`. (Thanks to Dan Luecking for pointing this out.) The argument #1 is the glossary type.

```
5224 \newcommand*{\@gls@glossary}[2]{%
5225   \if@gls@debug
5226     \PackageInfo{glossaries}{wrglossary(#1)(#2)}%
5227   \fi
5228   \index{#2}%
5229 }
```

This is a convenience command to set `\@gls@glossary`. It's used by `\@makeglossary` and then redefined to do nothing, as it only needs to be done once.

`s@renewglossary`

```
5230 \newcommand{\@gls@renewglossary}{%
5231   \gdef\@gls@glossary##1{\@bsphack\begingroup\gls@wrglossary{##1}}%
5232   \let\@gls@renewglossary\empty
5233 }
```

The `\gls@wrglossary` command is defined to have two arguments. The first argument is the glossary type, the second argument is the glossary entry (the format of which is set in `\glslink`).

`\gls@wrglossary`

```
5234 \newcommand*{\gls@wrglossary}[2]{%
5235   \ifglssavewrites
5236     \protected@edef\@gls@tmp{\the\csname glo@#1@filetok\endcsname#2}%
5237     \expandafter\global\expandafter\csname glo@#1@filetok\endcsname
5238       \expandafter{\@gls@tmp^^J}%
5239   \else
5240     \ifcsdef{glo@#1@file}%
5241       {%
5242         \expandafter\protected@write\csname glo@#1@file\endcsname{%
```

```

5243      \gls@disablepagerefexpansion}{#2}%
5244  }%
5245  {%
5246      \ifignoredglossary{#1}{}%
5247      {%
5248          \GlossariesWarning{No file defined for glossary ‘#1’}%
5249      }%
5250  }%
5251  \fi
5252  \endgroup\@esphack
5253 }

```

\do@wrglossary

```

5254 \newcommand*\do@wrglossary}[1]{%
5255   \glswriteentry{#1}{\do@wrglossary{#1}}%
5256 }

```

\glswriteentry Provide a user level command so the user can customize whether or not a line should be added to the glossary. The arguments are the label and the code that writes to the glossary file.

```

5257 \newcommand*\glswriteentry}[2]{%
5258   \ifglsindexonlyfirst
5259     \ifglsused{#1}{}{#2}%
5260   \else
5261     #2%
5262   \fi
5263 }

```

protected@pagefmts List of page formats to be protected against expansion.

```

5264 \newcommand{\gls@protected@pagefmts}{%
5265   \gls@numberpage,\gls@alphpage,\gls@Alphpage,\gls@romanpage,\gls@Romanpage,\gls@arabicpage%
5266 }

```

agerefexpansion

```

5267 \newcommand*\gls@disablepagerefexpansion){%
5268   \@for\@gls@this:=\gls@protected@pagefmts\do
5269   {%
5270     \expandafter\let\@gls@this\relax
5271   }%
5272 }

```

\gls@alphpage

```

5273 \newcommand*\gls@alphpage{\@alph\c@page}

```

\gls@Alphpage

```

5274 \newcommand*\gls@Alphpage{\@Alph\c@page}

```

\gls@numberpage

```

5275 \newcommand*\gls@numberpage{\number\c@page}

```

`\gls@arabicpage`

```
5276 \newcommand*{\gls@arabicpage}{\@arabic\c@page}
```

`\gls@romanpage`

```
5277 \newcommand*{\gls@romanpage}{\romannumeral\c@page}
```

`\gls@Romanpage`

```
5278 \newcommand*{\gls@Romanpage}{\@Roman\c@page}
```

`protectedpagefmt`

```
\glsaddprotectedpagefmt{<cs name>}
```

Added a page format to the list of protected page formats. The argument should be the name (without a backslash) of the command that takes a \TeX register as the argument (`\<csname>\c@page` must be valid).

```
5279 \newcommand*{\glsaddprotectedpagefmt}[1]{%
5280   \eappto\gls@protected@pagefmts{,\expandonce{\csname gls#1page\endcsname}}%
5281   \csedef{gls#1page}{\expandonce{\csname#1\endcsname}\noexpand\c@page}%
5282   \eappto\@wrglossarynumberhook{%
5283     \noexpand\let\expandonce{\csname org@gls#1\endcsname}%
5284     \expandonce{\csname#1\endcsname}%
5285     \noexpand\def\expandonce{\csname#1\endcsname}{%
5286       \noexpand\@wrglossary@pageformat
5287       \expandonce{\csname gls#1page\endcsname}%
5288       \expandonce{\csname org@gls#1\endcsname}%
5289     }%
5290   }%
5291 }
```

`ssarynumberhook` Hook used by `\@do@wrglossary`

```
5292 \newcommand*\@wrglossarynumberhook{}
```

`sary@pageformat`

```
5293 \newcommand{\@wrglossary@pageformat}[3]{%
5294   \ifx#3\c@page #1\else #2#3\fi
5295 }
```

`owprimitivemods` Conditional to determine whether or not `\@do@wrglossary` should be allowed to temporarily redefine `\the` and `\number`.

```
5296 \newif\ifglswrallowprimitivemods
5297 \glswrallowprimitivemodstrue
```

`@do@wrglossary` Write the glossary entry in the appropriate format. (Need to set `\@glsnumberformat` and `\@gls@counter` prior to use.) The argument is the entry's label.

```
5298 \newcommand*{\@do@wrglossary}[1]{%
5299   \begingroup
```


First a bit of hackery to prevent premature expansion of \c@page. Store original definitions:

```

5300 \let\orgthe\the
5301 \let\orgnumber\number

5302 \let\orgarabic\@arabic
5303 \let\orgromannumeral\romannumeral
5304 \let\orgalph\@alph
5305 \let\orgAlph\@Alph
5306 \let\orgRoman\@Roman

```

Redefine:

```

5307 \ifglswrallowprimitivemods
5308 \def\the##1{%
5309 \ifx##1\c@page \gls@numberpage\else\orgthe##1\fi}%
5310 \def\number##1{%
5311 \ifx##1\c@page \gls@numberpage\else\orgnumber##1\fi}%
5312 \fi
5313 \def\@arabic##1{%
5314 \ifx##1\c@page \gls@arabicpage\else\orgarabic##1\fi}%
5315 \def\romannumeral##1{%
5316 \ifx##1\c@page \gls@romanpage\else\orgromannumeral##1\fi}%
5317 \def\@Roman##1{%
5318 \ifx##1\c@page \gls@Romanpage\else\orgRoman##1\fi}%
5319 \def\@alph##1{%
5320 \ifx##1\c@page \gls@alphpage\else\orgalph##1\fi}%
5321 \def\@Alph##1{%
5322 \ifx##1\c@page \gls@Alphpage\else\orgAlph##1\fi}%

```

Add hook to allow for other number formats:

```

5323 \@wrglossarynumberhook

```

Prevent expansion:

```

5324 \gls@disablepagerefexpansion

```

Now store location in \glslocref:

```

5325 \protected@xdef\glslocref{\theHglentrycounter}%
5326 \endgroup

```

Escape any special characters

```

5327 \@gls@checkmkidxchars\glslocref

```

Check if the hyper-location is the same as the location and set the hyper prefix.

```

5328 \expandafter\ifx\theHglentrycounter\theHglentrycounter\relax
5329 \def\@glo@counterprefix{%
5330 \else
5331 \protected@edef\@glsHlocref{\theHglentrycounter}%
5332 \@gls@checkmkidxchars\@glsHlocref
5333 \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
5334 {\@glslocref}{\@glsHlocref}%
5335 }%
5336 \@do@gls@getcounterprefix
5337 \fi

```

De-tok label if required

```
5338 \edef\@gls@label{\glsdetoklabel{#1}}%
```

Write the information to file:

```
5339 \@do@@wrglossary
5340 }
```

@do@@wrglossary

```
5341 \newcommand*{\@do@@wrglossary}{%
```

Determine whether to use xindy or makeindex syntax

```
5342 \ifglsxindy
```

Need to determine if the formatting information starts with a (or) indicating a range.

```
5343 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
5344 \def\@glo@range{}%
5345 \expandafter\if\@glo@prefix(\relax
5346 \def\@glo@range{:open-range}%
5347 \else
5348 \expandafter\if\@glo@prefix)\relax
5349 \def\@glo@range{:close-range}%
5350 \fi
5351 \fi
```

Write to the glossary file using xindy syntax.

```
5352 \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
5353 (indexentry :key (\csname glo@\@gls@label @index\endcsname)
5354 :locref \string"\@glo@counterprefix}{\@glslocref}\string" %
5355 :attr \string"\@gls@counter\@glo@suffix\string"
5356 \@glo@range
5357 )
5358 }%
5359 \else
```

Convert the format information into the format required for makeindex

```
5360 \@set@glo@numformat{\@glo@numfmt}{\@gls@counter}{\@glsnumberformat}%
5361 {\@glo@counterprefix}%
```

Write to the glossary file using makeindex syntax.

```
5362 \gls@glossary{\csname glo@\@gls@label @type\endcsname}{%
5363 \string\glossaryentry{\csname glo@\@gls@label @index\endcsname
5364 \@gls@encapchar\@glo@numfmt}{\@glslocref}}%
5365 \fi
5366 }
```

etcounterprefix Get the prefix that needs to be prepended to counter in order to get the hyper counter. (For example, with the standard article class and hyperref, \theequation needs to be prefixed with <section num>. to get the equivalent \theHequation.) NB this assumes that the prefix ends with a dot, which is the standard. (Otherwise it makes the xindy location classes more complicated.)

```

5367 \newcommand*\@gls@getcounterprefix[2]{%
5368   \edef\@gls@thisloc{#1}\edef\@gls@thisHloc{#2}%
5369   \ifx\@gls@thisloc\@gls@thisHloc
5370     \def\@glo@counterprefix{}%
5371   \else
5372     \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
5373       \def\@glo@tmp{##2}%
5374       \ifx\@glo@tmp\@empty
5375         \def\@glo@counterprefix{}%
5376       \else
5377         \def\@glo@counterprefix{##1}%
5378       \fi
5379     }%
5380   \@gls@get@counterprefix#2.#1\end@getprefix

Warn if no prefix can be formed.

5381   \ifx\@glo@counterprefix\@empty
5382     \GlossariesWarning{Hyper target ‘#2’ can’t be formed by
5383       prefixing^^Jlocation ‘#1’. You need to modify the
5384       definition of \string\theH\@gls@counter^^Jotherwise you
5385       will get the warning: “‘name{\@gls@counter.#1}’ has been^^J
5386       referenced but does not exist”}%
5387   \fi
5388 \fi
5389 }

```

1.15 Glossary Entry Cross-References

`\do@seeglossary` Write the glossary entry with a cross reference. The first argument is the entry’s label, the second must be in the form `[\langle tag \rangle]{\langle list \rangle}`, where `\langle tag \rangle` is a tag such as “see” and `\langle list \rangle` is a list of labels.

```

5390 \newcommand{\do@seeglossary}[2]{%
5391   \def\@gls@xref{#2}%
5392   \@onelevel@sanitize\@gls@xref
5393   \@gls@checkmkidxchars\@gls@xref
5394   \ifglxsindy
5395     \gls@glossary{\csname glo@#1@type\endcsname}{%
5396       (indexentry
5397         :tkey (\csname glo@#1@index\endcsname)
5398         :xref (\string"\@gls@xref\string")
5399         :attr \string"see\string"
5400       )
5401     }%
5402   \else
5403     \gls@glossary{\csname glo@#1@type\endcsname}{%
5404       \string\glossaryentry{\csname glo@#1@index\endcsname
5405         \@gls@encapchar glsseeformat\@gls@xref}{Z}}%
5406   \fi

```

5407 }

`\@gls@fixbraces` If no optional argument is specified, list needs to be enclosed in a set of braces.

```
5408 \def\@gls@fixbraces#1#2#3\@nil{%
5409   \ifx#2[\relax
5410     \@gls@fixbraces#1#2#3\@end@fixbraces
5411   \else
5412     \def#1{{#2#3}}%
5413   \fi
5414 }
```

`@@gls@fixbraces`

```
5415 \def\@@gls@fixbraces#1[#2]#3\@end@fixbraces{%
5416   \def#1{[#2]{#3}}%
5417 }
```

`\glssee` `\glssee{<label>}{<cross-ref list>}`

```
5418 \DeclareRobustCommand*\glssee[3][\seename]{%
5419   \@do@seeglossary{#2}{#1}{#3}}
5420 \newcommand*\@glssee[3][\seename]{%
5421   \glssee[#1]{#3}{#2}}
```

`\glsseeformat` The first argument specifies what tag to use (e.g. “see”), the second argument is a comma-separated list of labels. The final argument (the location) is ignored.

```
5422 \DeclareRobustCommand*\glsseeformat[3][\seename]{%
5423   \emph{#1} \glsseelist{#2}}
```

`\glsseelist` `\glsseelist{<list>}` formats list of entry labels.

```
5424 \DeclareRobustCommand*\glsseelist[1]{%
```

If there is only one item in the list, set the last separator to do nothing.

```
5425   \let\@gls@dolast\relax
```

Don’t display separator on the first iteration of the loop

```
5426   \let\@gls@donext\relax
```

Iterate through the labels

```
5427   \@for\@gls@thislabel:=#1\do{%
```

Check if on last iteration of loop

```
5428     \ifx\@xfor@nextelement\@nnil
5429       \@gls@dolast
5430     \else
5431       \@gls@donext
5432     \fi
```

Display the entry for this label. (Expanding label as it’s a temporary control sequence that’s used elsewhere.)

```
5433     \expandafter\glsseeitem\expandafter{\@gls@thislabel}%
```

Update separators

```
5434 \let\@gls@dolast\glsseelastsep
5435 \let\@gls@donext\glsseesep
5436 }%
5437 }
```

`\glsseelastsep` Separator to use between penultimate and ultimate entries in a cross-referencing list.

```
5438 \newcommand*{\glsseelastsep}{\space\andname\space}
```

`\glsseesep` Separator to use between entires in a cross-referencing list.

```
5439 \newcommand*{\glsseesep}{, }
```

`\glsseeitem` `\glsseeitem{<label>}` formats individual entry in a cross-referencing list.

```
5440 \DeclareRobustCommand*{\glsseeitem}[1]{\glshyperlink[\glsseeitemformat{#1}]{#1}}
```

`\glsseeitemformat` As from v3.0, default is to use `\glentrytext` instead of `\glentryname`. (To avoid problems with the name key being sanitized.)

```
5441 \newcommand*{\glsseeitemformat}[1]{\glentrytext{#1}}
```

1.16 Displaying the glossary

An individual glossary is displayed in the text using `\printglossary[<key-val list>]`. If the type key is omitted, the default glossary is displayed. The optional argument can be used to specify an alternative glossary, and can also be used to set the style, title and entry in the table of contents. Available keys are defined below.

`\save@numberlist` Provide command to store number list.

```
5442 \newcommand*{\gls@save@numberlist}[1]{%
5443   \ifglssavenumberlist
5444     \toks@{#1}%
5445     \edef\@do@writeaux@info{%
5446       \noexpand\csgdef{glo@\glscurrententrylabel @numberlist}{\the\toks@}%
5447     }%
5448     \@onelevel@sanitize\@do@writeaux@info
5449     \protected@write\@auxout{}{\@do@writeaux@info}%
5450   \fi
5451 }
```

`\noprintglossary` Warn the user if they have forgotten `\printglossaries` or `\printglossary`. (Will be suppressed if there is at least one occurrence of `\printglossary`. There is no check to ensure that there is a `\printglossary` for each defined glossary.)

```
5452 \newcommand*{\warn@noprintglossary}{}%
```

`\printglossary` The TOC title needs to be processed in a different manner to the main title in case the translator and hyperref packages are both being used.

```
5453 \ifcsundef{printglossary}{}%
5454 {%
```

If `\printglossary` is already defined, issue a warning and undefine it.

```
5455 \@gls@warnonglossdefined
5456 \undef\printglossary
5457 }
```

`\printglossary` has an optional argument. The default value is to set the glossary type to the main glossary.

```
5458 \newcommand*{\printglossary}[1][type=\glsdefaulttype]{%
5459 \@printglossary{#1}{\@print@glossary}%
5460 }
```

The `\printglossaries` command will do `\printglossary` for each glossary type that has been defined. It is better to use `\printglossaries` rather than individual `\printglossary` commands to ensure that you don't forget any new glossaries you may have created. It also makes it easier to chop and change the value of the acronym package option. However, if you want to list the glossaries in a different order, or if you want to set the title or table of contents entry, or if you want to use different glossary styles for each glossary, you will need to use `\printglossary` explicitly for each glossary type.

`\printglossaries`

```
5461 \newcommand*{\printglossaries}{%
5462 \forallglossaries{\@glo@type}{\printglossary[type=\@glo@type]}%
5463 }
```

`\printnoidxglossary` Provide an alternative to `\printglossary` that doesn't require an external indexing application. Entries won't be sorted and the location list will be empty.

```
5464 \newcommand*{\printnoidxglossary}[1][type=\glsdefaulttype]{%
5465 \@printglossary{#1}{\@print@noidx@glossary}%
5466 }
```

`\printnoidxglossaries` Analogous to `\printglossaries`

```
5467 \newcommand*{\printnoidxglossaries}{%
5468 \forallglossaries{\@glo@type}{\printnoidxglossary[type=\@glo@type]}%
5469 }
```

`\printgloss@setsort` Initialise to do nothing.

```
5470 \newcommand*{\@printgloss@setsort}{}%
```

`\preglossaryhook`

```
5471 \newcommand*{\@gls@preglossaryhook}{}%
```

`\@printglossary` Sets up the glossary for either `\printglossary` or `\printnoidxglossary`. The first argument is the options list, the second argument is the handler macro that deals with the actual glossary.

```
5472 \newcommand{\@printglossary}[2]{%
5473 \def\@glo@type{\glsdefaulttype}%
5474 \def\glossarytitle{\csname @glo@type@\@glo@type @title\endcsname}%
5475 }
```

Set up defaults.

```
5473 \def\@glo@type{\glsdefaulttype}%
5474 \def\glossarytitle{\csname @glo@type@\@glo@type @title\endcsname}%
5475 }
```

```

5475 \def\glossarytoctitle{\glossarytitle}%
5476 \let\org@glossarytitle\glossarytitle

5477 \def\@glossarystyle{%
5478   \ifx\@glossary@default@style\relax
5479     \GlossariesWarning{No default glossary style provided \MessageBreak
5480       for the glossary '\@glo@type'. \MessageBreak
5481       Using deprecated fallback. \MessageBreak
5482       To fix this set the style with \MessageBreak
5483       \string\setglossarystyle\space or use the \MessageBreak
5484       style key=value option}%
5485   \fi
5486 }%
5487 \def\gls@dotoc@title{\gls@set@title{\@glo@type}}%

  Store current value of \glossaryentrynumbers. (This may be changed via the optional ar-
  gument)
5488 \let\@org@glossaryentrynumbers\glossaryentrynumbers

  Localise the effects of the optional argument
5489 \bgroup

  Activate or deactivate sort key:
5490 \@printgloss@setsort

  Determine settings specified in the optional argument.
5491 \setkeys{printgloss}{#1}%

  If title has been set, but toctitle hasn't, make toctitle the same as given title (rather than the
  title used when the glossary was defined)
5492 \ifx\glossarytitle\org@glossarytitle
5493 \else
5494   \expandafter\let\csname @glo@type@\@glo@type @title\endcsname
5495     \glossarytitle
5496 \fi

  Allow a high-level user command to indicate the current glossary
5497 \let\currentglossary\@glo@type

  Enable individual number lists to be suppressed.
5498 \let\org@glossaryentrynumbers\glossaryentrynumbers
5499 \let\glsnonextpages\@glsnonextpages

  Enable individual number list to be activated:
5500 \let\glsnextpages\@glsnextpages

  Enable suppression of description terminators.
5501 \let\nopostdesc\@nopostdesc

  Set up the entry for the TOC
5502 \gls@dotoc@title

  Set the glossary style
5503 \@glossarystyle

```

Added a way to fetch the current entry label (v3.08 updated for new `\glossentry` and `\subglossentry`, but this is now only needed for backward compatibility):

```

5504 \let\gls@org@glossaryentryfield\glossentry
5505 \let\gls@org@glossarysubentryfield\subglossentry
5506 \renewcommand{\glossentry}[1]{%
5507   \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
5508   \gls@org@glossaryentryfield{##1}%
5509 }%
5510 \renewcommand{\subglossentry}[2]{%
5511   \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
5512   \gls@org@glossarysubentryfield{##1}{##2}%
5513 }%
5514 \@gls@preglossaryhook

```

Now do the handler macro that deals with the actual glossary:

```

5515   #2%
    End the current scope
5516 \egroup
    Reset \glossaryentrynumbers
5517 \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
    Suppress warning about no \printglossary
5518 \global\let\warn@noprntglossary\relax
5519 }

```

`@print@glossary` Internal workings of `\printglossary` dealing with reading the external file.

```

5520 \newcommand{\@print@glossary}{%

```

Some macros may end up being expanded into internals in the glossary, so need to make `@` a letter. (Unlikely to be a problem since v3.08a but kept for backward compatibility.)

```

5521 \makeatletter

```

Input the glossary file, if it exists.

```

5522 \@input@{\jobname.\csname @glotype@\@glo@type @in\endcsname}%

```

If the glossary file doesn't exist, do `\null`. (This ensures that the page is shipped out and all write commands are done.) This might produce an empty page, but at this point the document isn't complete, so it shouldn't matter.

```

5523 \IfFileExists{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
5524 {}%
5525 {\null}%

```

If `xindy` is being used, need to write the language dependent information to the `.aux` file for `makeglossaries`.

```

5526 \ifglsxindy
5527   \ifcsundef{@xdy@\@glo@type @language}%
5528   {%
5529     \edef\@do@auxoutstuff{%
5530       \noexpand\AtEndDocument{%

```


If the user removes the glossary package from their document, ensure the next run doesn't throw a load of undefined control sequence errors when the aux file is parsed.

```

5531      \noexpand\immediate\noexpand\write\@auxout{%
5532      \string\providecommand\string\@xdylanguage[2]{}}%
5533      \noexpand\immediate\noexpand\write\@auxout{%
5534      \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
5535      }%
5536    }%
5537  }%
5538  {%
5539    \edef\@do@auxoutstuff{%
5540    \noexpand\AtEndDocument{%
5541      \noexpand\immediate\noexpand\write\@auxout{%
5542      \string\providecommand\string\@xdylanguage[2]{}}%
5543      \noexpand\immediate\noexpand\write\@auxout{%
5544      \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
5545      @language\endcsname}}%
5546    }%
5547  }%
5548  }%
5549  \@do@auxoutstuff
5550  \edef\@do@auxoutstuff{%
5551  \noexpand\AtEndDocument{%

```

If the user removes the glossaries package from their document, ensure the next run doesn't throw a load of undefined control sequence errors when the aux file is parsed.

```

5552      \noexpand\immediate\noexpand\write\@auxout{%
5553      \string\providecommand\string\@gls@codepage[2]{}}%
5554      \noexpand\immediate\noexpand\write\@auxout{%
5555      \string\@gls@codepage{\@glo@type}{\@gls@codepage}}%
5556    }%
5557  }%
5558  \@do@auxoutstuff
5559 \fi

```

Activate warning if \makeglossaries hasn't been used.

```

5560 \renewcommand*{\@warn@nomakeglossaries}{%
5561   \GlossariesWarningNoLine{\string\makeglossaries\space
5562   hasn't been used,^^Jthe glossaries will not be updated}%
5563 }%
5564 }

```

The sort macros all have the syntax:

`\@glo@sortmacro@<order>{<type>}`

where *<order>* is the sort order as specified by the sort key and *<type>* is the glossary type. (The referenced entry list is stored in \@glsref@<type>. The actual sorting is done by \@glo@sortentries{<handler>}{<type>}).

glo@sortentries

```
5565 \newcommand*{\@glo@sortentries}[2]{%
5566   \def\@glo@sortinglist{}%
5567   \def\@glo@sortinghandler{#1}%
5568   \edef\@glo@type{#2}%
5569   \forlistcsloop{\@glo@do@sortentries}{\@glsref@#2}%
5570   \csdef{\@glsref@#2}{}%
5571   \@for\@this@label:=\@glo@sortinglist\do{%
      Has this entry already been added?
5572     \xifinlistcs{\@this@label}{\@glsref@#2}%
5573     {}%
5574     {%
5575       \listcsxadd{\@glsref@#2}{\@this@label}%
5576     }%
5577     \ifcsdef{\@glo@sortingchildren@\@this@label}%
5578     {%
5579       \@glo@addchildren{#2}{\@this@label}%
5580     }%
5581     {}%
5582   }%
5583 }
```

@glo@addchildren

`\@glo@addchildren{<type>}{<parent>}`

```
5584 \newcommand*{\@glo@addchildren}[2]{%
```

Scope to allow nesting.

```
5585   \bgroup
5586     \letcs{\@glo@childlist}{\@glo@sortingchildren@#2}%
5587     \@for\@this@childlabel:=\@glo@childlist\do
5588     {%
```

Check this label hasn't already been added.

```
5589       \xifinlistcs{\@this@childlabel}{\@glsref@#1}%
5590       {}%
5591       {%
5592         \listcsxadd{\@glsref@#1}{\@this@childlabel}%
5593       }%
```

Does this child have children?

```
5594       \ifcsdef{\@glo@sortingchildren@\@this@childlabel}%
5595       {%
5596         \@glo@addchildren{#1}{\@this@childlabel}%
5597       }%
5598       {%
5599       }%
5600     }%
5601   \egroup
5602 }
```

@do@sortentries

```
5603 \newcommand*{\@glo@do@sortentries}[1]{%
5604   \ifglshasparent{#1}%
5605   {%
```

This entry has a parent, so add it to the child list

```
5606   \edef\@glo@parent{\csuse{glo@glstetoklabel{#1}@parent}}%
5607   \ifcsundef{glo@sortingchildren@\@glo@parent}%
5608   {%
5609     \csdef{glo@sortingchildren@\@glo@parent}{}%
5610   }%
5611   {}%
5612   \expandafter\@glo@sortedinsert
5613     \csname @glo@sortingchildren@\@glo@parent\endcsname{#1}%
```

Has the parent been added?

```
5614   \xifinlistcs{\@glo@parent}{@glsref@\@glo@type}%
5615   {%
```

Yes, it has so do nothing.

```
5616   }%
5617   {%
```

No, it hasn't so add it now.

```
5618     \expandafter\@glo@do@sortentries\expandafter{\@glo@parent}%
5619     }%
5620   }%
5621   {%
5622     \@glo@sortedinsert{\@glo@sortinglist}{#1}%
5623   }%
5624 }
```

glo@sortedinsert

`\@glo@sortedinsert{<list>}{<entry label>}`

Insert into list.

```
5625 \newcommand*{\@glo@sortedinsert}[2]{%
5626   \dtl@insertinto{#2}{#1}{\@glo@sortinghandler}%
5627 }
```

The sort handlers need to be in the form required by datatool's `\dtl@sortlist` macro. These must set the count register `\dtl@sortresult` to either -1 ($\#1$ less than $\#2$), 0 ($\#1 = \#2$) or $+1$ ($\#1$ greater than $\#2$).

orthandler@word

```
5628 \newcommand*{\@glo@sorthandler@word}[2]{%
5629   \letcs@gls@sort@A{glo@glstetoklabel{#1}@sort}%
5630   \letcs@gls@sort@B{glo@glstetoklabel{#2}@sort}%
5631   \edef@glo@do@compare{%
5632     \noexpand\dtlwordindexcompare{\noexpand\dtl@sortresult}%
```

```

5633     {\expandonce\@gls@sort@B}%
5634     {\expandonce\@gls@sort@A}%
5635 }%
5636 \glo@do@compare
5637 }

```

thandler@letter

```

5638 \newcommand*{\@glo@sorthandler@letter}[2]{%
5639   \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5640   \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5641   \edef\glo@do@compare{%
5642     \noexpand\dtlletterindexcompare{\noexpand\dtl@sortresult}%
5643     {\expandonce\@gls@sort@B}%
5644     {\expandonce\@gls@sort@A}%
5645   }%
5646   \glo@do@compare
5647 }

```

orthandler@case Case-sensitive sort.

```

5648 \newcommand*{\@glo@sorthandler@case}[2]{%
5649   \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5650   \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5651   \edef\glo@do@compare{%
5652     \noexpand\dtlcompare{\noexpand\dtl@sortresult}%
5653     {\expandonce\@gls@sort@B}%
5654     {\expandonce\@gls@sort@A}%
5655   }%
5656   \glo@do@compare
5657 }

```

thandler@nocase Case-insensitive sort.

```

5658 \newcommand*{\@glo@sorthandler@nocase}[2]{%
5659   \letcs\@gls@sort@A{glo\glsdetoklabel{#1}@sort}%
5660   \letcs\@gls@sort@B{glo\glsdetoklabel{#2}@sort}%
5661   \edef\glo@do@compare{%
5662     \noexpand\dtlicompare{\noexpand\dtl@sortresult}%
5663     {\expandonce\@gls@sort@B}%
5664     {\expandonce\@gls@sort@A}%
5665   }%
5666   \glo@do@compare
5667 }

```

@sortmacro@word Sort macro for ‘word’

```

5668 \newcommand*{\@glo@sortmacro@word}[1]{%
5669   \ifdefstring{\@glo@default@sorttype}{standard}%
5670   {%
5671     \@glo@sortentries{\@glo@sorthandler@word}{#1}%
5672   }%
5673   {%

```

```

5674 \PackageError{glossaries}{Conflicting sort options:^^J
5675 \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5676 \string\printnoidxglossary[sort=word]}{}%
5677 }%
5678 }

```

ortmacro@letter Sort macro for ‘letter’

```

5679 \newcommand*{\@glo@sortmacro@letter}[1]{%
5680 \ifdefstring{\@glo@default@sorttype}{standard}%
5681 {%
5682 \@glo@sortentries{\@glo@sorthandler@letter}{#1}%
5683 }%
5684 {%
5685 \PackageError{glossaries}{Conflicting sort options:^^J
5686 \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5687 \string\printnoidxglossary[sort=letter]}{}%
5688 }%
5689 }

```

tmacro@standard Sort macro for ‘standard’. (Use either ‘word’ or ‘letter’ order.)

```

5690 \newcommand*{\@glo@sortmacro@standard}[1]{%
5691 \ifdefstring{\@glo@default@sorttype}{standard}%
5692 {%
5693 \ifcsdef{\@glo@sorthandler@glorder}%
5694 {%
5695 \@glo@sortentries{\csuse{\@glo@sorthandler@glorder}}{#1}%
5696 }%
5697 {%
5698 \PackageError{glossaries}{Unknown sort handler ‘\glorder’}{}%
5699 }%
5700 }%
5701 {%
5702 \PackageError{glossaries}{Conflicting sort options:^^J
5703 \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5704 \string\printnoidxglossary[sort=standard]}{}%
5705 }%
5706 }

```

@sortmacro@case Sort macro for ‘case’

```

5707 \newcommand*{\@glo@sortmacro@case}[1]{%
5708 \ifdefstring{\@glo@default@sorttype}{standard}%
5709 {%
5710 \@glo@sortentries{\@glo@sorthandler@case}{#1}%
5711 }%
5712 {%
5713 \PackageError{glossaries}{Conflicting sort options:^^J
5714 \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5715 \string\printnoidxglossary[sort=case]}{}%
5716 }%

```

5717 }

ortmacro@nocase Sort macro for ‘nocase’

```
5718 \newcommand*{\@glo@sortmacro@nocase}[1]{%
5719   \ifdefstring{\@glo@default@sorttype}{standard}%
5720   {%
5721     \@glo@sortentries{\@glo@sorthandler@nocase}{#1}%
5722   }%
5723   {%
5724     \PackageError{glossaries}{Conflicting sort options:^^J
5725       \string\usepackage[sort=\@glo@default@sorttype]{glossaries}^^J
5726       \string\printnoidxglossary[sort=nocase]}{}%
5727   }%
5728 }
```

o@sortmacro@def Sort macro for ‘def’. The order of definition is given in \glo@list@*type*.

```
5729 \newcommand*{\@glo@sortmacro@def}[1]{%
5730   \def\@glo@sortinglist{%
5731     \for@gl@sentries[#1]{\@gls@thislabel}%
5732     {%
5733       \xifinlistcs{\@gls@thislabel}{\@gls@sref@#1}%
5734       {%
5735         \list@add{\@glo@sortinglist}{\@gls@thislabel}%
5736       }%
5737     }%
5738   }%
5739   }%
5740   \cslet{\@gls@sref@#1}{\@glo@sortinglist}%
5741 }
```

Hasn't been referenced.

```
5738   }%
5739   }%
5740   \cslet{\@gls@sref@#1}{\@glo@sortinglist}%
5741 }
```

ortmacro@def@do This won't include parent entries that haven't been referenced.

```
5742 \newcommand*{\@glo@sortmacro@def@do}[1]{%
5743   \ifinlistcs{#1}{\@gls@sref@\@glo@type}%
5744   {}%
5745   {%
5746     \listcsadd{\@gls@sref@\@glo@type}{#1}%
5747   }%
5748   \ifcsdef{\@glo@sortingchildren@#1}%
5749   {%
5750     \@glo@addchildren{\@glo@type}{#1}%
5751   }%
5752   {}%
5753 }
```

o@sortmacro@use Sort macro for ‘use’. (No sorting is required, as the entries are already in order of use, so do nothing.)

```
5754 \newcommand*{\@glo@sortmacro@use}[1]{}
```

@noidx@glossary Glossary handler for \printnoidxglossary which doesn't use an indexing application. Since \printnoidxglossary may occur at the start of the document, we can't just check if an entry has been used. Instead, the first pass needs to write information to the aux file every time an entry is referenced. This needs to be read in on the second run and stored in a list corresponding to the appropriate glossary.

```
5755 \newcommand*{\@print@noidx@glossary}{%
5756   \ifcsdef{@glsref@{\@glo@type}}%
5757   {%
```

Sort the entries:

```
5758   \ifcsdef{@glo@sortmacro@{\@glo@sorttype}}%
5759   {%
5760     \csuse{@glo@sortmacro@{\@glo@sorttype}}{\@glo@type}%
5761   }%
5762   {%
5763     \PackageError{glossaries}{Unknown sort handler '@glo@sorttype'}{ }%
5764   }%
```

Do the glossary heading and preamble

```
5765   \glossarysection[\glossarytoctitle]{\glossarytitle}%
5766   \glossarypreamble
```

The glossary style might use a tabular-like environment, which may cause scoping problems when setting the current letter group. The predefined tabular-like styles don't support letter group headings, but there's nothing to stop the user from defining their own custom style that might, so any redefinition of this command within theglossary will have to be done globally.

```
5767   \def\@gls@currentlettergroup{}%
5768   \begin{theglossary}%
5769   \glossaryheader
5770   \glsresetentrylist
```

Iterate through the entries.

```
5771   \forlistcsloop{\@gls@noidx@do}{@glsref@{\@glo@type}}%
```

Finally end the glossary and do the postamble:

```
5772   \end{theglossary}%
5773   \glossarypostamble
5774 }%
5775 {%
5776   \@gls@noref@warn{\@glo@type}%
5777 }%
5778 }
```

\glo@grabfirst

```
5779 \def\glo@grabfirst#1#2\@nil{%
5780   \def\@gls@firsttok{#1}%
5781   \ifdefempty\@gls@firsttok
5782   {%
5783     \def\@glo@thislettergrp{0}%
5784   }%
5785   {%
```

Sanitize it:

```
5786 \@onelevel@sanitize\@gls@firsttok
```

Fetch the first letter:

```
5787 \expandafter\@glo@grabfirst\@gls@firsttok{}\{}\@nil
5788 }%
5789 }
```

\@glo@grabfirst

```
5790 \def\@glo@grabfirst#1#2\@nil{%
5791 \ifdefempty\@glo@thislettergrp
5792 {%
5793 \def\@glo@thislettergrp{glssymbols}%
5794 }%
5795 {%
5796 \count@=\uccode'#1\relax
5797 \ifnum\count@=0\relax
5798 \def\@glo@thislettergrp{glssymbols}%
5799 \else
5800 \ifdefstring\@glo@sorttype{case}%
5801 {%
5802 \count@='#1\relax
5803 }%
5804 {%
5805 }%
5806 \edef\@glo@thislettergrp{\the\count@}%
5807 \fi
5808 }%
5809 }
```

\@gls@noidx@do Handler for list iteration used by \@print@noidx@glossary. The argument is the entry label.
This only allows one sublevel.

```
5810 \newcommand{\@gls@noidx@do}[1]{%
```

Get this entry's location list

```
5811 \global\letcs{\@gls@loclist}{\glo@\glsdetoklabel{#1}@loclist}%
```

Does this entry have a parent?

```
5812 \ifglshasparent{#1}%
5813 {%
```

Has a parent.

```
5814 \gls@level=\csuse{\glo@\glsdetoklabel{#1}@level}\relax
5815 \ifdefvoid{\@gls@loclist}
5816 {%
5817 \subglossentry{\gls@level}{#1}{}%
5818 }%
5819 {%
5820 \subglossentry{\gls@level}{#1}%
5821 }
```



```

5822     \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}}%
5823   }%
5824 }%
5825 }%
5826 {%

```

Doesn't have a parent Get this entry's sort key

```

5827   \letcs{\@gls@sort}{\glo@\glsdetoklabel{#1}@sort}%

```

Fetch the first letter:

```

5828   \expandafter\glo@grabfirst\@gls@sort{}\@nil
5829   \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
5830   {}%
5831   {%

```

Do the group header:

```

5832     \ifdefempty{\@gls@currentlettergroup}{\@gls@groupskip}%
5833     \gls@groupheading{\@glo@thislettergrp}%
5834   }%

```

```

5835   \global\let\@gls@currentlettergroup\@glo@thislettergrp

```

Do this entry:

```

5836   \ifdefvoid{\@gls@loclist}
5837   {%
5838     \glossentry{#1}{}%
5839   }%
5840   {%
5841     \glossentry{#1}%
5842   }%
5843     \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}}%
5844   }%
5845 }%
5846 }%
5847 }

```

\glsnoidxloclist \glsnoidxloclist{<list cs>}

Display location list.

```

5848 \newcommand*{\glsnoidxloclist}[1]{%
5849   \def\@gls@noidxloclist@sep{}%
5850   \def\@gls@noidxloclist@prev{}%
5851   \forlistloop{\glsnoidxloclisthandler}{#1}%
5852 }

```

\loclisthandler Handler for location list iterator.

```

5853 \newcommand*{\glsnoidxloclisthandler}[1]{%
5854   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5855   {%

```

Same as previous location so skip.

```

5856 }%
5857 {%
5858   \@gls@noidxloclist@sep
5859   #1%
5860   \def\@gls@noidxloclist@sep{\delimN}%
5861   \def\@gls@noidxloclist@prev{#1}%
5862 }%
5863 }

```

`\loclisthandler` Handler for location list iterator when used with `\glsdisplaynumberlist`.

```

5864 \newcommand*\glsnoidxdisplayloclisthandler[1]{%
5865   \ifdefstring{\@gls@noidxloclist@prev}{#1}%
5866   {%

```

Same as previous location so skip.

```

5867 }%
5868 {%
5869   \@gls@noidxloclist@sep
5870   \@gls@noidxloclist@prev
5871   \def\@gls@noidxloclist@prev{#1}%
5872 }%
5873 }

```

`\noidxdisplayloc` `\glsnoidxdisplayloc{<prefix>}{<counter>}{<format>}{<location>}`

Display a location in the location list.

```

5874 \newcommand*\glsnoidxdisplayloc[4]{%
5875   \setentrycounter[#1]{#2}%
5876   \csuse{#3}{#4}%
5877 }

```

`\@gls@reference` `\@gls@reference{<type>}{<label>}{<loc>}`

Identifies that a reference has been used (for use in the aux file). All entries must be defined in the preamble.

```

5878 \newcommand*\@gls@reference[3]{%

```

Add to label list

```

5879   \glsdoifexistsorwarn{#2}%
5880   {%
5881     \ifcsundef{\@glsref@#1}{\csgdef{\@glsref@#1}{}}{}%
5882     \ifinlistcs{#2}{\@glsref@#1}%
5883     {}%
5884     {\listcsgadd{\@glsref@#1}{#2}}%

```

Add to location list

```

5885 \ifcsundef{glo@glstetoklabel{#2}@loclist}%
5886 {\csgdef{glo@glstetoklabel{#2}@loclist}{}}%
5887 {}%
5888 \listcsgadd{glo@glstetoklabel{#2}@loclist}{#3}%
5889 }%
5890 }

```

The keys that can be used in the optional argument to `\printglossary` or `\printnoidxglossary` are as follows: The type key sets the glossary type.

```

5891 \define@key{printgloss}{type}{\def@glo@type{#1}}

```

The title key sets the title used in the glossary section header. This overrides the title used in `\newglossary`.

```

5892 \define@key{printgloss}{title}{%
5893 \def@glossarytitle{#1}%
5894 \let@gls@dotoc@title\relax
5895 }

```

The toctitle sets the text used for the relevant entry in the table of contents.

```

5896 \define@key{printgloss}{toctitle}{%
5897 \def@glossarytoctitle{#1}%
5898 \let@gls@dotoc@title\relax
5899 }

```

The style key sets the glossary style (but only for the given glossary).

```

5900 \define@key{printgloss}{style}{%
5901 \ifcsundef{@glsstyle@#1}%
5902 {%
5903 \PackageError{glossaries}%
5904 {Glossary style ‘#1’ undefined}{}%
5905 }%
5906 {%
5907 \def@glossarystyle{\setglossentrycompatibility
5908 \csname @glsstyle@#1\endcsname}%
5909 }%
5910 }

```

The numberedsection key determines if this glossary should be in a numbered section.

```

5911 \define@choicekey{printgloss}{numberedsection}[\val\nr]{%
5912 false,nolabel,autolabel,nameref}[nolabel]{%
5913 \ifcase\nr\relax
5914 \renewcommand*{\@glossarysecstar}{*}%
5915 \renewcommand*{\@glossaryseclabel}{}%
5916 \or
5917 \renewcommand*{\@glossarysecstar}{}%
5918 \renewcommand*{\@glossaryseclabel}{}%
5919 \or
5920 \renewcommand*{\@glossarysecstar}{}%
5921 \renewcommand*{\@glossaryseclabel}{\label{glsautoprefix@glo@type}}%

```

```

5922 \or
5923   \renewcommand*{\@@glossarysecstar}{*}%
5924   \renewcommand*{\@@glossaryseclabel}{%
5925     \protected@edef\@currentlabelname{\glossarytoctitle}%
5926     \label{\glsautoprefix\@glo@type}}%
5927 \fi
5928 }

```

The `nogroupskip` key determines whether or not there should be a vertical gap between glossary groups.

```

5929 \define@choicekey{printgloss}{nogroupskip}{true,false}[true]{%
5930   \csuse{glsnogroupskip#1}%
5931 }

```

The `nopostdot` key has the same effect as the package option of the same name.

```

5932 \define@choicekey{printgloss}{nopostdot}{true,false}[true]{%
5933   \csuse{glsnopostdot#1}%
5934 }

```

The `entrycounter` key is the same as the package option but localised to the current glossary.

```

5935 \define@choicekey{printgloss}{entrycounter}{true,false}[true]{%
5936   \csuse{glsentrycounter#1}%
5937   \ifglsentrycounter
5938     \ifx\@gls@counterwithin\@empty
5939       \newcounter{glossaryentry}%
5940     \else
5941       \newcounter{glossaryentry}[\@gls@counterwithin]%
5942     \fi
5943     \def\theHglossaryentry{\currentglossary.\theglossaryentry}%
5944     \renewcommand*{\glsresetentrycounter}{%
5945       \setcounter{glossaryentry}{0}%
5946     }%
5947     \renewcommand*{\glsstepentry}[1]{%
5948       \refstepcounter{glossaryentry}%
5949       \label{glsentry-\glsdetoklabel{##1}}%
5950     }%
5951     \renewcommand*{\glsentrycounterlabel}{\theglossaryentry.\space}%
5952     \renewcommand*{\glsentryitem}[1]{%
5953       \glsstepentry{##1}\glsentrycounterlabel
5954     }%
5955   \else
5956     \renewcommand*{\glsresetentrycounter}{}%
5957     \renewcommand*{\glsstepentry}[1]{}%
5958     \renewcommand*{\glsentrycounterlabel}{}%
5959     \renewcommand*{\glsentryitem}[1]{\glsresetsubentrycounter}
5960 \fi
5961 }

```

The `subentrycounter` key is the same as the package option but localised to the current glossary. Note that this doesn't affect the master/slave counter attributes, which occurs if `subentrycounter` and `entrycounter` package options are set to true.

```

5962 \define@choicekey{printgloss}{subentrycounter}{true,false}[true]{%
5963   \csuse{glssubentrycounter#1}%
5964   \ifglssubentrycounter
5965     \ifundef\c@glossarysubentry
5966     {%
5967       \ifglsentrycounter
5968       \newcounter{glossarysubentry}[glossaryentry]%
5969     \else
5970       \newcounter{glossarysubentry}
5971     \fi
5972   }{}%
5973   \renewcommand*{\glssubentry}[1]{%
5974     \edef\currentglssubentry{\glsetoklabel{##1}}%
5975     \refstepcounter{glossarysubentry}%
5976     \label{glentry-\currentglssubentry}%
5977   }%
5978   \renewcommand*{\glsresetsubentrycounter}{%
5979     \setcounter{glossarysubentry}{0}%
5980   }%
5981   \renewcommand*{\glssubentryitem}[1]{%
5982     \glssubentry{##1}\glssubentrycounterlabel
5983   }%
5984   \renewcommand*{\glssubentrycounterlabel}{\theglossarysubentry\space}%
5985   \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
5986 \else
5987   \renewcommand*{\glssubentryitem}[1]{}%
5988   \renewcommand*{\glssubentry}[1]{}%
5989   \renewcommand*{\glsresetsubentrycounter}{}%
5990   \renewcommand*{\glssubentrycounterlabel}{}%
5991 \fi
5992 }

```

The nonumberlist key determines if this glossary should have a number list.

```

5993 \define@boolkey{printgloss}[gls]{nonumberlist}[true]{%
5994 \ifglsnonumberlist
5995   \def\glossaryentrynumbers##1{%
5996 \else
5997   \def\glossaryentrynumbers##1{##1}%
5998 \fi}

```

The sort key sets the glossary sort handler (\printnoidxglossary only).

```

5999 \define@key{printgloss}{sort}{\@glo@assign@sortkey{#1}}

```

@assign@sortkey Issue error if used with \printglossary

```

6000 \newcommand*{\@glo@no@assign@sortkey}[1]{%
6001   \PackageError{glossaries}{‘sort’ key not permitted with
6002     \string\printglossary}%
6003   {The ‘sort’ key may only be used with \string\printnoidxglossary}%
6004 }

```

@assign@sortkey For use with `\printnoidxglossary`

```

6005 \newcommand*{\@glo@assign@sortkey}[1]{%
6006   \def\@glo@sorttype{#1}%
6007 }

```

\glsnonextpages Suppresses the next number list only. Global assignments required as it may not occur in the same level of grouping as the next numberlist. (For example, if `\glsnonextpages` is place in the entry's description and 3 column tabular style glossary is used.) `\org@glossaryentrynumbers` needs to be set at the start of each glossary, in the event that `\glossaryentrynumber` is re-defined.

```

6008 \newcommand*{\@glsnonextpages}{%
6009   \gdef\glossaryentrynumbers##1{%
6010     \glsresetentrylist
6011   }%
6012 }

```

\glsnextpages Activate the next number list only. Global assignments required as it may not occur in the same level of grouping as the next numberlist. (For example, if `\glsnextpages` is place in the entry's description and 3 column tabular style glossary is used.) `\org@glossaryentrynumbers` needs to be set at the start of each glossary, in the event that `\glossaryentrynumber` is re-defined.

```

6013 \newcommand*{\@glsnextpages}{%
6014   \gdef\glossaryentrynumbers##1{%
6015     ##1\glsresetentrylist}}

```

glsresetentrylist Resets `\glossaryentrynumbers`

```

6016 \newcommand*{\glsresetentrylist}{%
6017   \global\let\glossaryentrynumbers\org@glossaryentrynumbers}

```

\glsnonextpages Outside of `\printglossary` this does nothing.

```

6018 \newcommand*{\glsnonextpages}{}

```

\glsnextpages Outside of `\printglossary` this does nothing.

```

6019 \newcommand*{\glsnextpages}{}

```

glossaryentry If the `entrycounter` package option has been used, define a counter to number each level 0 entry.

```

6020 \ifglseentrycounter
6021   \ifx\@gls@counterwithin\@empty
6022     \newcounter{glossaryentry}
6023   \else
6024     \newcounter{glossaryentry}[\@gls@counterwithin]
6025   \fi
6026   \def\theHglossaryentry{\currentglossary.\theglossaryentry}
6027 \fi

```

`\glossarysubentry` If the `subentrycounter` package option has been used, define a counter to number each level 1 entry.

```

6028 \ifglssubentrycounter
6029   \ifglsentrycounter
6030     \newcounter{glossarysubentry}[glossaryentry]
6031   \else
6032     \newcounter{glossarysubentry}
6033   \fi
6034   \def\theHglossarysubentry{\currentglssubentry.\theglossarysubentry}
6035 \fi

```

`subentrycounter` Resets the `glossarysubentry` counter.

```

6036 \ifglssubentrycounter
6037   \newcommand*{\glsresetsubentrycounter}{%
6038     \setcounter{glossarysubentry}{0}%
6039   }
6040 \else
6041   \newcommand*{\glsresetsubentrycounter}{}
6042 \fi

```

`subentrycounter` Resets the `glossentry` counter.

```

6043 \ifglsentrycounter
6044   \newcommand*{\glsresetentrycounter}{%
6045     \setcounter{glossaryentry}{0}%
6046   }
6047 \else
6048   \newcommand*{\glsresetentrycounter}{}
6049 \fi

```

`\glsstepentry` Advance the `glossaryentry` counter if in use. The argument is the label associated with the entry.

```

6050 \ifglsentrycounter
6051   \newcommand*{\glsstepentry}[1]{%
6052     \refstepcounter{glossaryentry}%
6053     \label{glsentry-\glsdetoklabel{#1}}%
6054   }
6055 \else
6056   \newcommand*{\glsstepentry}[1]{}
6057 \fi

```

`glsstepsubentry` Advance the `glossarysubentry` counter if in use. The argument is the label associated with the subentry.

```

6058 \ifglssubentrycounter
6059   \newcommand*{\glsstepsubentry}[1]{%
6060     \edef\currentglssubentry{\glsdetoklabel{#1}}%
6061     \refstepcounter{glossarysubentry}%
6062     \label{glsentry-\currentglssubentry}%
6063   }

```

```

6064 \else
6065   \newcommand*{\glsstepsubentry}[1]{%
6066 \fi

```

`\glsrefentry` Reference the entry or sub-entry counter if in use, otherwise just do `\gls`.

```

6067 \ifglssentrycounter
6068   \newcommand*{\glsrefentry}[1]{\ref{glssentry-\glsdetoklabel{#1}}}
6069 \else
6070   \ifglssubentrycounter
6071     \newcommand*{\glsrefentry}[1]{\ref{glssentry-\glsdetoklabel{#1}}}
6072   \else
6073     \newcommand*{\glsrefentry}[1]{\gls{#1}}
6074   \fi
6075 \fi

```

`trycounterlabel` Defines how to display the glossaryentry counter.

```

6076 \ifglssentrycounter
6077   \newcommand*{\glsentrycounterlabel}{\theglossaryentry.\space}
6078 \else
6079   \newcommand*{\glsentrycounterlabel}{}
6080 \fi

```

`trycounterlabel` Defines how to display the glossarysubentry counter.

```

6081 \ifglssubentrycounter
6082   \newcommand*{\glssubentrycounterlabel}{\theglossarysubentry)\space}
6083 \else
6084   \newcommand*{\glssubentrycounterlabel}{}
6085 \fi

```

`\glsentryitem` Step and display glossaryentry counter, if appropriate.

```

6086 \ifglssentrycounter
6087   \newcommand*{\glsentryitem}[1]{%
6088     \glsstepentry{#1}\glsentrycounterlabel
6089   }
6090 \else
6091   \newcommand*{\glsentryitem}[1]{\glsresetsubentrycounter}
6092 \fi

```

`glssubentryitem` Step and display glossarysubentry counter, if appropriate.

```

6093 \ifglssubentrycounter
6094   \newcommand*{\glssubentryitem}[1]{%
6095     \glsstepsubentry{#1}\glssubentrycounterlabel
6096   }
6097 \else
6098   \newcommand*{\glssubentryitem}[1]{}
6099 \fi

```

`theglossary` If the `theglossary` environment has already been defined, a warning will be issued. This environment should be redefined by glossary styles.


```

6100 \ifcsundef{theglossary}%
6101 {%
6102   \newenvironment{theglossary}{}{}%
6103 }%
6104 {%
6105   \@gls@warnontheGLOSSdefined
6106   \renewenvironment{theglossary}{}{}%
6107 }

```

The glossary header is given by `\glossaryheader`. This forms part of the glossary style, and must indicate what should appear immediately after the start of the `theglossary` environment. (For example, if the glossary uses a tabular-like environment, it may be used to set the header row.) Note that if you don't want a header row, the glossary style must redefine `\glossaryheader` to do nothing.

`\glossaryheader`

```

6108 \newcommand*{\glossaryheader}{}

```

`\glstarget` `\glstarget{<label>}{<name>}`

Provide user interface to `\@glstarget` to make it easier to modify the glossary style in the document.

```

6109 \newcommand*{\glstarget}[2]{\@glstarget{\glo@linkprefix#1}{#2}}

```

As from version 3.08, glossary information is now written to the external files using `\glossentry` and `\subglossentry` instead of `\glossaryentryfield` and `\glossarysubentryfield`. The default definition provides backward compatibility for glossary styles that use the old forms.

`\compatibleglossentry`

```
\glossentry{<label>}{<page-list>}
```

```

6110 \providecommand*{\compatibleglossentry}[2]{%
6111   \toks0{#2}%
6112   \protected@edef\@do@glossentry{\noexpand\glossaryentryfield{#1}%
6113     {\noexpand\glsnamefont
6114       {\expandafter\expandonce\csname glo@#1@name\endcsname}}}%
6115     {\expandafter\expandonce\csname glo@#1@desc\endcsname}%
6116     {\expandafter\expandonce\csname glo@#1@symbol\endcsname}%
6117     {\the\toks0}}%
6118   }%
6119   \@do@glossentry
6120 }

```

`\glossentryname`

```

6121 \newcommand*{\glossentryname}[1]{%
6122   \glsdoifexistsorwarn{#1}%
6123   {%

```

```

6124 \letcs{\glo@name}{\glo@\glsdetoklabel{#1}@name}%
6125 \expandafter\glsnamefont\expandafter{\glo@name}%
6126 }%
6127 }

```

\Glossentryname

```

6128 \newcommand*{\Glossentryname}[1]{%
6129 \glsdoifexistsorwarn{#1}%
6130 {%
6131 \glsnamefont{\Glsentryname{#1}}%
6132 }%
6133 }

```

\glossentrydesc

```

6134 \newcommand*{\glossentrydesc}[1]{%
6135 \glsdoifexistsorwarn{#1}%
6136 {%
6137 \glentrydesc{#1}%
6138 }%
6139 }

```

\Glossentrydesc

```

6140 \newcommand*{\Glossentrydesc}[1]{%
6141 \glsdoifexistsorwarn{#1}%
6142 {%
6143 \Glsentrydesc{#1}%
6144 }%
6145 }

```

lossentrysymbol

```

6146 \newcommand*{\glossentrysymbol}[1]{%
6147 \glsdoifexistsorwarn{#1}%
6148 {%
6149 \glentrysymbol{#1}%
6150 }%
6151 }

```

lossentrysymbol

```

6152 \newcommand*{\Glossentrysymbol}[1]{%
6153 \glsdoifexistsorwarn{#1}%
6154 {%
6155 \Glsentrysymbol{#1}%
6156 }%
6157 }

```

blesubglossentry

```
\subglossentry{<level>}{<label>}{<page-list>}
```

```

6158 \providecommand*\compatiblesubglossentry}[3]{%
6159   \toks@{#3}%
6160   \protected@edef\@do@subglossentry{\noexpand\glossarysubentryfield{\number#1}%
6161     {#2}%
6162     {\noexpand\glsnamefont
6163       {\expandafter\expandonce\csname glo@#2@name\endcsname}}}%
6164     {\expandafter\expandonce\csname glo@#2@desc\endcsname}%
6165     {\expandafter\expandonce\csname glo@#2@symbol\endcsname}%
6166     {\the\toks@}%
6167   }%
6168   \@do@subglossentry
6169 }

```

rycompatibility

```

6170 \newcommand*\setglossentrycompatibility{%
6171   \let\glossentry\compatibleglossentry
6172   \let\subglossentry\compatiblesubglossentry
6173 }
6174 \setglossentrycompatibility

```

glossaryentryfield

```

\glossaryentryfield{<label>}{<name>}{<description>}{<symbol>}
{<page-list>}

```

This command formerly governed how each entry row should be formatted in the glossary. Now deprecated.

```

6175 \newcommand{\glossaryentryfield}[5]{%
6176   \GlossariesWarning
6177   {Deprecated use of \string\glossaryentryfield.^^J
6178     I recommend you change to \string\glossentry.^^J
6179     If you've just upgraded, try removing your gls auxiliary
6180     files^^J and recompile}%
6181   \noindent\textbf{\glstarget{#1}{#2}} #4 #3. #5\par}

```

glossarysubentryfield

```

\glossarysubentryfield{<level>}{<label>}{<name>}{<description>}{<symbol>}
{<page-list>}

```

This command governs how each subentry should be formatted in the glossary. Glossary styles need to redefine this command. Most of the predefined styles ignore *<symbol>*. The first argument is a number indicating the level. (The level should be greater than or equal to 1.)

```

6182 \newcommand*\glossarysubentryfield}[6]{%
6183   \GlossariesWarning
6184   {Deprecated use of \string\glossarysubentryfield.^^J
6185     I recommend you change to \string\subglossentry.^^J
6186     If you've just upgraded, try removing your gls auxiliary

```

```

6187   files^^J and recompile}%
6188   \glstarget{#2}{\strut}#4. #6\par}

```

Within each glossary, the entries form distinct groups which are determined by the first character of the sort key. When using `makeindex`, there will be a maximum of 28 groups: symbols, numbers, and the 26 alphabetical groups A, ..., Z. If you use `xindy` the groups will depend on whatever alphabet is used. This is determined by the language or custom alphabets can be created in the `xindy` style file. The command `\glsgroupskip` specifies what to do between glossary groups. Glossary styles must redefine this command. (Note that `\glsgroupskip` only occurs between groups, not at the start or end of the glossary.)

`\glsgroupskip`

```

6189 \newcommand*{\glsgroupskip}{}

```

Each of the 28 glossary groups described above is preceded by a group heading. This is formatted by the command `\glsgroupheading` which takes one argument which is the *label* assigned to that group (not the title). The corresponding labels are: `glssymbols`, `glsnumbers`, A, ..., Z. Glossary styles must redefine this command. (In between groups, `\glsgroupheading` comes immediately after `\glsgroupskip`.)

`\glsgroupheading`

```

6190 \newcommand*{\glsgroupheading}[1]{}

```

It is possible to “trick” `makeindex` into treating entries as though they belong to the same group, even if the terms don’t start with the same letter, by modifying the sort key. For example, all entries belonging to one group could be defined so that the sort key starts with an a, while entries belonging to another group could be defined so that the sort key starts with a b, and so on. If you want each group to have a heading, you would then need to modify the translation control sequences `\glsgetgrouptitle` and `\glsgetgrouplabel` so that the label is translated into the required title (and vice-versa).

`\glsgetgrouptitle{<label>}`

This command produces the title for the glossary group whose label is given by *<label>*. By default, the group labelled `glssymbols` produces `\glssymbolsgroupname`, the group labelled `glsnumbers` produces `\glsnumbersgroupname` and all the other groups simply produce their label. As mentioned above, the group labels are: `glssymbols`, `glsnumbers`, A, ..., Z. If you want to redefine the group titles, you will need to redefine this command. Languages other than English may produce labels that are non-expandable, so we need to check for that otherwise it will create a “missing `\endcsname` inserted” error.

`\glsgetgrouptitle`

```

6191 \newcommand*{\glsgetgrouptitle}[1]{%
6192   \@gls@getgrouptitle{#1}{\@gls@grptitle}%
6193   \@gls@grptitle
6194 }

```

`\gls@getgrouptitle` Gets the group title specified by the label (first argument) and stores in the second argument, which must be a control sequence.

```

6195 \newcommand*{\@gls@getgrouptitle}[2]{%
    Even if the argument appears to be a single letter, it won't be considered a single letter by
    \dtl@ifsingle if it's an active character.
6196 \dtl@ifsingle{#1}%
6197 {%
6198   \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6199 }%
6200 {%
6201   \ifboolexpr{test{\ifstrequal{#1}{glssymbols}}
6202               or test{\ifstrequal{#1}{glsnumbers}}}%
6203   {%
6204     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6205   }%
6206   {%
6207     \def#2{#1}%
6208   }%
6209 }%
6210 }

```

`\othergrouptitle` Version for the no-indexing app option:

```

6211 \newcommand*{\@gls@noidx@getgrouptitle}[2]{%
6212   \DTLifint{#1}%
6213   {\edef#2{\char#1\relax}}%
6214   {%
6215     \ifcsundef{#1groupname}{\def#2{#1}}{\letcs#2{#1groupname}}%
6216   }%
6217 }

```

`\glsgetgrouplabel{<title>}`

This command does the reverse to the previous command. The argument is the group title, and it produces the group label. Note that if you redefine `\glsgetgrouptitle`, you will also need to redefine `\glsgetgrouplabel`.

`\glsgetgrouplabel`

```

6218 \newcommand*{\glsgetgrouplabel}[1]{%
6219 \ifthenelse{\equal{#1}{\glssymbolsgroupname}}{\glssymbols}{%
6220 \ifthenelse{\equal{#1}{\glsnumbersgroupname}}{\glsnumbers}{#1}}

```

The command `\setentrycounter` sets the entry's associated counter (required by `\glshypernumber` etc.) `\glslink` and `\glsadd` encode the `\glossary` argument so that the relevant counter is set prior to the formatting command.

`\setentrycounter`

```

6221 \newcommand*{\setentrycounter}[2][ ]{%

```

```

6222 \def\@glo@counterprefix{#1}%
6223 \ifx\@glo@counterprefix\empty
6224   \def\@glo@counterprefix{.}%
6225 \else
6226   \def\@glo@counterprefix{.#1.}%
6227 \fi
6228 \def\glsentrycounter{#2}%
6229 }

```

The current glossary style can be set using `\setglossarystyle{<style>}`.

`\setglossarystyle`

```

6230 \newcommand*{\setglossarystyle}[1]{%
6231   \ifcsundef{@glsstyle@#1}%
6232   {%
6233     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
6234   }%
6235   {%
6236     \csname @glsstyle@#1\endcsname
6237   }%

```

Set the default style if it's not already set.

```

6238 \ifx\@glossary@default@style\relax
6239   \protected@edef\@glossary@default@style{#1}%
6240 \fi
6241 }

```

`\glossarystyle`

```

6242 \newcommand*{\glossarystyle}[1]{%
6243   \ifcsundef{@glsstyle@#1}%
6244   {%
6245     \PackageError{glossaries}{Glossary style ‘#1’ undefined}{}%
6246   }%
6247   {%
6248     \GlossariesWarning
6249     {Deprecated command \string\glossarystyle.^^J
6250     I recommend you switch to \string\setglossarystyle\space unless
6251     you want to maintain backward compatibility}%
6252     \setglossentrycompatibility
6253     \csname @glsstyle@#1\endcsname

6254     \ifcsdef{@glscompstyle@#1}%
6255     {\setglossentrycompatibility\csuse{@glscompstyle@#1}}%
6256     {}%
6257   }%

```

Set the default style if it isn't already set so that `\printglossary` can warn if the fallback style is in use.

```

6258 \ifx\@glossary@default@style\relax
6259   \protected@edef\@glossary@default@style{#1}%

```

```

6260 \fi
6261 }

```

`\newglossarystyle` New glossary styles can be defined using:

```
\newglossarystyle{<name>}{<definition>}
```

The *<definition>* argument should redefine `\theglossary`, `\glossaryheader`, `\glsgroupheading`, `\glossaryentryfield` and `\glsgroupskip` (see [section 1.19](#) for the definitions of predefined styles). Glossary styles should not redefine `\glossarypreamble` and `\glossarypostamble`, as the user should be able to switch between styles without affecting the pre- and postambles.

```

6262 \newcommand{\newglossarystyle}[2]{%
6263   \ifcsundef{@glsstyle@#1}%
6264   {%
6265     \expandafter\def\csname @glsstyle@#1\endcsname{#2}%
6266   }%
6267   {%
6268     \PackageError{glossaries}{Glossary style ‘#1’ is already defined}{}%
6269   }%
6270 }

```

`\newglossarystyle` Code for this macro supplied by Marco Daniel.

```

6271 \newcommand{\renewglossarystyle}[2]{%
6272   \ifcsundef{@glsstyle@#1}%
6273   {%
6274     \PackageError{glossaries}{Glossary style ‘#1’ isn’t already defined}{}%
6275   }%
6276   {%
6277     \csdef{@glsstyle@#1}{#2}%
6278   }%
6279 }

```

Glossary entries are encoded so that the second argument to `\glossaryentryfield` is always specified as `\glsnamefont{<name>}`. This allows the user to change the font used to display the name term without having to redefine `\glossaryentryfield`. The default uses the surrounding font, so in the list type styles (which place the name in the optional argument to `\item`) the name will appear in bold.

`\glsnamefont`

```
\newcommand*{\glsnamefont}[1]{#1}
```

Each glossary entry has an associated number list (usually page numbers) that indicate where in the document the entry has been used. The format for these number lists can be changed using the format key in commands like `\glslink`. The default format is given by `\glsnumber`. This takes a single argument which may be a single number, a number range or a number list. The number ranges are delimited with `\delimR`, the number lists are delimited with `\delimN`.

If the document doesn't have hyperlinks, the numbers can be displayed just as they are, but if the document supports hyperlinks, the numbers should link to the relevant location. This means extracting the individual numbers from the list or ranges. The package does this with the `\hyperpage` command, but this is encoded for comma and dash delimiters and only for the page counter, but this code needs to be more general. So I have adapted the code used in the package.

`\glshypernumber`

```
6281 \ifcsundef{hyperlink}%
6282 {%
6283   \def\glshypernumber#1{#1}%
6284 }%
6285 {%
6286   \def\glshypernumber#1{\@glshypernumber#1\nohyperpage{}}\@nil}
6287 }
```

`\@glshypernumber` This code was provided by Heiko Oberdiek to allow material to be attached to the location.

```
6288 \def\@glshypernumber#1\nohyperpage#2#3\@nil{%
6289   \ifx\#1\%
6290     \else
6291       \@delimR#1\delimR\delimR\%
6292     \fi
6293     \ifx\#2\%
6294       \else
6295         #2%
6296       \fi
6297       \ifx\#3\%
6298         \else
6299           \@glshypernumber#3\@nil
6300         \fi
6301 }
```

`\@delimR` displays a range of numbers for the counter whose name is given by `\@gls@counter` (which must be set prior to using `\glshypernumber`).

`\@delimR`

```
6302 \def\@delimR#1\delimR #2\delimR #3\{%
6303 \ifx\#2\%
6304   \@delimN{#1}%
6305 \else
6306   \@gls@numberlink{#1}\delimR\@gls@numberlink{#2}%
6307 \fi}
```

`\@delimN` displays a list of individual numbers, instead of a range:

`\@delimN`

```
6308 \def\@delimN#1{\@delimN#1\delimN \delimN\%
6309 \def\@delimN#1\delimN #2\delimN#3\{%
6310 \ifx\#3\%
```



```

6311 \@gls@numberlink{#1}%
6312 \else
6313 \@gls@numberlink{#1}\delimN\@gls@numberlink{#2}%
6314 \fi
6315 }

```

The following code is modified from hyperref's \HyInd@pagelink where the name of the counter being used is given by \@gls@counter.

```

6316 \def\@gls@numberlink#1{%
6317 \begingroup
6318 \toks@={}%
6319 \@gls@removespaces#1 \@nil
6320 \endgroup}

6321 \def\@gls@removespaces#1 #2\@nil{%
6322 \toks@=\expandafter{\the\toks@#1}%
6323 \ifx\#2\%
6324 \edef\x{\the\toks@}%
6325 \ifx\x\empty
6326 \else

6327 \hyperlink{\glstrycounter\@glo@counterprefix\the\toks@}%
6328 {\the\toks@}%
6329 \fi
6330 \else
6331 \@gls@ReturnAfterFi{%
6332 \@gls@removespaces#2\@nil
6333 }%
6334 \fi
6335 }
6336 \long\def\@gls@ReturnAfterFi#1\fi{\fi#1}

```

The following commands will switch to the appropriate font, and create a hyperlink, if hyperlinks are supported. If hyperlinks are not supported, they will just display their argument in the appropriate font.

```

\hyperrm
6337 \newcommand*\hyperrm[1]{\textrm{\glshypernumber{#1}}}

\hypersf
6338 \newcommand*\hypersf[1]{\textsf{\glshypernumber{#1}}}

\hypertt
6339 \newcommand*\hypertt[1]{\texttt{\glshypernumber{#1}}}

\hyperbf
6340 \newcommand*\hyperbf[1]{\textbf{\glshypernumber{#1}}}

\hypermd
6341 \newcommand*\hypermd[1]{\textmd{\glshypernumber{#1}}}

```

`\hyperit`

```
6342 \newcommand*{\hyperit}[1]{\textit{\glshypernumber{#1}}}
```

`\hypersl`

```
6343 \newcommand*{\hypersl}[1]{\textsl{\glshypernumber{#1}}}
```

`\hyperup`

```
6344 \newcommand*{\hyperup}[1]{\textup{\glshypernumber{#1}}}
```

`\hypersc`

```
6345 \newcommand*{\hypersc}[1]{\textsc{\glshypernumber{#1}}}
```

`\hyperemph`

```
6346 \newcommand*{\hyperemph}[1]{\emph{\glshypernumber{#1}}}
```

1.17 Acronyms

`\oldacronym` `\oldacronym[<label>]{<abbrv>}{<long>}{<key-val list>}`

This emulates the way the old package defined acronyms. It is equivalent to `\newacronym[<key-val list>]{<label>}{<abbrv>}{<long>}` and it additionally defines the command `\<label>` which is equivalent to `\gls{<label>}` (thus *<label>* must only contain alphabetical characters). If *<label>* is omitted, *<abbrv>* is used. This only emulates the syntax of the old package. The way the acronyms appear in the list of acronyms is determined by the definition of `\newacronym` and the glossary style.

Note that `\<label>` can't have an optional argument if the package is loaded. If hasn't been loaded then you can do `\<label>[<insert>]` but you can't do `\<label>[<key-val list>]`. For example if you define the acronym `svm`, then you can do `\svm['s]` but you can't do `\svm[format=textbf]`. If the package is loaded, `\svm['s]` will appear as `svm ['s]` which is unlikely to be the desired result. In this case, you will need to use `\gls` explicitly, e.g. `\gls{svm}['s]`. Note that it is up to the user to load if desired.

```
6347 \newcommand{\oldacronym}[4][\gls@label]{%
6348   \def\gls@label{#2}%
6349   \newacronym[#4]{#1}{#2}{#3}%
6350   \ifcsundef{xspace}%
6351   {%
6352     \expandafter\edef\csname#1\endcsname{%
6353       \noexpand\@ifstar{\noexpand\Gls{#1}}{\noexpand\gls{#1}}}%
6354   }%
6355 }%
6356 {%
6357   \expandafter\edef\csname#1\endcsname{%
6358     \noexpand\@ifstar{\noexpand\Gls{#1}\noexpand\xspace}{%
6359       \noexpand\gls{#1}\noexpand\xspace}%

```

```

6360     }%
6361   }%
6362 }

```

```
\newacronym[⟨key-val list⟩]{⟨label⟩}{⟨abbrev⟩}{⟨long⟩}
```

This is a quick way of defining acronyms, using `\newglossaryentry` with the appropriate values. It sets the glossary type to `\acronymtype` which will be `acronym` if the package option `acronym` has been used, otherwise it will be the default glossary. Since `\newacronym` merely calls `\newglossaryentry`, the acronym is treated like any other glossary entry.

If you prefer a different format, you can redefine `\newacronym` as required. The optional argument can be used to override any of the settings.

This is just a stub. It's redefined by commands like `\SetDefaultAcronymStyle`.

`\newacronym`

```
6363 \newcommand{\newacronym}[4] [] {}
```

Set up some convenient short cuts. These need to be changed if `\newacronym` is changed (or if the description key is changed).

`\acrpluralsuffix` Plural suffix used by `\newacronym`. This just defaults to `\glspluralsuffix` but is changed to include `\textup` if the `smallcaps` option is used, so that the suffix doesn't appear in small caps as it doesn't look right. For example, `ABCS` looks as though the “s” is part of the acronym, but `ABCS` looks as though the “s” is a plural suffix. Since the entire text `abcs` is set in `\textsc`, `\textup` is needed to cancel it out.

```
6364 \newcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}
```

If `garamondx` has been loaded, need to use `\textulc` instead of `\textup`.

`\glstextup`

```
6365 \newrobustcmd*{\glstextup}[1]{\ifdef\textulc{\textulc{#1}}{\textup{#1}}}
```

The following are defined for compatibility with version 2.07 and earlier.

`\glsshortkey`

```
6366 \newcommand*{\glsshortkey}{short}
```

`\glsshortpluralkey`

```
6367 \newcommand*{\glsshortpluralkey}{shortplural}
```

`\glslongkey`

```
6368 \newcommand*{\glslongkey}{long}
```

`\glslongpluralkey`

```
6369 \newcommand*{\glslongpluralkey}{longplural}
```

`\acrfull` Full form of the acronym.

```
6370 \newrobustcmd*{\acrfull}{\@gls@hyp@opt\ns@acrfull}

6371 \newcommand*\ns@acrfull[2] [] {%
6372   \new@ifnextchar[{\@acrfull{#1}{#2}}%
6373     {\@acrfull{#1}{#2} []}%
6374 }
```

`\@acrfull` Low-level macro:

```
6375 \def\@acrfull#1#2[#3] {%
    Make it easier for acronym styles to change this:
6376   \acrfullfmt{#1}{#2}{#3}%
6377 }
```

Using `\acrlinkfullformat` and `\acrfullformat` is now deprecated as it can cause complications with the first letter upper case variants, but the package needs to provide backward compatibility support.

`\acrfullfmt` No case change full format.

```
6378 \newcommand*\acrfullfmt[3] {%
6379   \acrlinkfullformat{\@acrlong}{\@acrshort}{#1}{#2}{#3}%
6380 }
```

`\acrlinkfullformat` Format for full links like `\acrfull`. Syntax: `\acrlinkfullformat{<long cs>}{<short cs>}{<options>}{<label>}{<insert>}`

```
6381 \newcommand{\acrlinkfullformat}[5] {%
6382   \acrfullformat{#1{#3}{#4}[#5]}{#2{#3}{#4} []}%
6383 }
```

`\acrfullformat` Default full form is `<long>` (`<short>`).

```
6384 \newcommand{\acrfullformat}[2]{#1\glsspace(#2)}
```

`\glsspace` Robust space to ensure it's written to the `.glsdefs` file.

```
6385 \newrobustcmd{\glsspace}{\space}
```

Default format for full acronym

`\Acrfull`

```
6386 \newrobustcmd*{\Acrfull}{\@gls@hyp@opt\ns@Acrfull}

6387 \newcommand*\ns@Acrfull[2] [] {%
6388   \new@ifnextchar[{\@Acrfull{#1}{#2}}%
6389     {\@Acrfull{#1}{#2} []}%
6390 }
```

Low-level macro:

```
6391 \def\@Acrfull#1#2[#3] {%
```

Make it easier for acronym styles to change this:

```
6392 \Acrfullfmt{#1}{#2}{#3}%  
6393 }
```

\Acrfullfmt First letter upper case full format.

```
6394 \newcommand*{\Acrfullfmt}[3]{%  
6395 \acrlinkfullformat{\@Acrlong}{\@acrshort}{#1}{#2}{#3}%  
6396 }
```

\ACRfull

```
6397 \newrobustcmd*{\ACRfull}{\@gls@hyp@opt\ns@ACRfull}  
  
6398 \newcommand*\ns@ACRfull[2][{}]{%  
6399 \new@ifnextchar[{\@ACRfull{#1}{#2}}%  
6400 {\@ACRfull{#1}{#2}[]}%  
6401 }
```

Low-level macro:

```
6402 \def\@ACRfull#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6403 \ACRfullfmt{#1}{#2}{#3}%  
6404 }
```

\ACRfullfmt All upper case full format.

```
6405 \newcommand*{\ACRfullfmt}[3]{%  
6406 \acrlinkfullformat{\@ACRlong}{\@ACRshort}{#1}{#2}{#3}%  
6407 }
```

Plural:

\acrfullpl

```
6408 \newrobustcmd*{\acrfullpl}{\@gls@hyp@opt\ns@acrfullpl}  
  
6409 \newcommand*\ns@acrfullpl[2][{}]{%  
6410 \new@ifnextchar[{\@acrfullpl{#1}{#2}}%  
6411 {\@acrfullpl{#1}{#2}[]}%  
6412 }
```

Low-level macro:

```
6413 \def\@acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6414 \acrfullplfmt{#1}{#2}{#3}%  
6415 }
```

\acrfullplfmt No case change plural full format.

```
6416 \newcommand*{\acrfullplfmt}[3]{%  
6417 \acrlinkfullformat{\@acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%  
6418 }
```

`\Acrfullpl`

```
6419 \newrobustcmd*{\Acrfullpl}{\@gls@hyp@opt\ns@Acrfullpl}
```

```
6420 \newcommand*{\ns@Acrfullpl}[2][\%  
6421   \new@ifnextchar[{\@Acrfullpl{#1}{#2}}%  
6422   {\@Acrfullpl{#1}{#2}[]}]%  
6423 }
```

Low-level macro:

```
6424 \def\@Acrfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6425   \Acrfullplfmt{#1}{#2}{#3}%  
6426 }
```

`\Acrfullplfmt` First letter upper case plural full format.

```
6427 \newcommand*{\Acrfullplfmt}[3]{%  
6428   \acrlinkfullformat{\@Acrlongpl}{\@acrshortpl}{#1}{#2}{#3}%  
6429 }
```

`\ACRfullpl`

```
6430 \newrobustcmd*{\ACRfullpl}{\@gls@hyp@opt\ns@ACRfullpl}
```

```
6431 \newcommand*{\ns@ACRfullpl}[2][\%  
6432   \new@ifnextchar[{\@ACRfullpl{#1}{#2}}%  
6433   {\@ACRfullpl{#1}{#2}[]}]%  
6434 }
```

Low-level macro:

```
6435 \def\@ACRfullpl#1#2[#3]{%
```

Make it easier for acronym styles to change this:

```
6436   \ACRfullplfmt{#1}{#2}{#3}%  
6437 }
```

`\ACRfullplfmt` All upper case plural full format.

```
6438 \newcommand*{\ACRfullplfmt}[3]{%  
6439   \acrlinkfullformat{\@ACRlongpl}{\@ACRshortpl}{#1}{#2}{#3}%  
6440 }
```

1.18 Predefined acronym styles

`\acronymfont` This is only used with the additional acronym styles:

```
6441 \newcommand{\acronymfont}[1]{#1}
```

`\firstacronymfont` This is only used with the additional acronym styles:

```
6442 \newcommand{\firstacronymfont}[1]{\acronymfont{#1}}
```

`\acronymformat` The styles that allow an additional description use `\acronymformat{<short>}{<long>}` to determine what information is displayed in the name.

```
6443 \newcommand*{\acronymformat}[2]{\acronymfont{#1}}
```

Define some tokens used by `\newacronym`:

`\glskeylisttok`

```
6444 \newtoks\glskeylisttok
```

`\glslabeltok`

```
6445 \newtoks\glslabeltok
```

`\glsshorttok`

```
6446 \newtoks\glsshorttok
```

`\glslongtok`

```
6447 \newtoks\glslongtok
```

`\newacronymhook` Provide a hook for `\newacronym`:

```
6448 \newcommand*{\newacronymhook}{}
```

`\genericNewAcronym` New improved version of setting the acronym style.

```
6449 \newcommand*{\SetGenericNewAcronym}{%
```

Change the behaviour of `\Glsentryname` to workaround expansion issues that cause a problem for `\makefirstuc`

```
6450 \let\@Gls@entryname\@Gls@acrentryname
```

Change the way acronyms are defined:

```
6451 \renewcommand{\newacronym}[4][\{%
6452   \ifdefempty{\@glsacronymlists}%
6453   {%
6454     \def\@glo@type{\acronymtype}%
6455     \setkeys{glossentry}{##1}%
6456     \DeclareAcronymList{\@glo@type}%
6457   }%
6458   }%
6459   \glskeylisttok{##1}%
6460   \glslabeltok{##2}%
6461   \glsshorttok{##3}%
6462   \glslongtok{##4}%
6463   \newacronymhook
6464   \protected@edef\@do@newglossaryentry{%
6465     \noexpand\newglossaryentry{\the\glslabeltok}%
6466     {%
6467       type=\acronymtype,%
6468       name={\expandonce{\acronymentry{##2}}},%
6469       sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
6470       text={\the\glsshorttok},%
```

```

6471      short={\the\glsshorttok},%
6472      shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6473      long={\the\glslongtok},%
6474      longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6475      \GenericAcronymFields,%
6476      \the\glskeylisttok
6477  }%
6478 }%
6479 \do@newglossaryentry
6480 }%

```

Make sure that \acrfull etc reflects the new style:

```

6481 \renewcommand*\acrfullfmt}[3]{%
6482   \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
6483 \renewcommand*\Acrfullfmt}[3]{%
6484   \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
6485 \renewcommand*\ACRfullfmt}[3]{%
6486   \glslink[##1]{##2}{%
6487     \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}%
6488 \renewcommand*\acrfullplfmt}[3]{%
6489   \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
6490 \renewcommand*\Acrfullplfmt}[3]{%
6491   \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
6492 \renewcommand*\ACRfullplfmt}[3]{%
6493   \glslink[##1]{##2}{%
6494     \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}%

```

Make sure that \glsentryfull etc reflects the new style:

```

6495 \renewcommand*\glsentryfull}[1]{\genacrfullformat{##1}{}}%
6496 \renewcommand*\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
6497 \renewcommand*\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
6498 \renewcommand*\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
6499 }

```

`\icAcronymFields` Fields used by \SetGenericNewAcronym that can be changed by the acronym style.

```

6500 \newcommand*\GenericAcronymFields{description={\the\glslongtok}}

```

`\acronymentry` `\acronymentry{<label>}`

Display style for the name field in the list of acronyms.

```

6501 \newcommand*\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}

```

`\acronymsort` `\acronymsort{<short>}{<long>}`

Default sort format for acronyms.

```

6502 \newcommand*\acronymsort}[2]{#1}

```


`\setacronymstyle` `\setacronymstyle{<style name>}`

```

6503 \newcommand*{\setacronymstyle}[1]{%
6504   \ifcsundef{@glsacr@dispstyle@#1}%
6505   {%
6506     \PackageError{glossaries}{Undefined acronym style ‘#1’}{}%
6507   }%
6508   {%
6509     \ifdefempty{@glsacronymlists}%
6510     {%
6511       \DeclareAcronymList{\acronymtype}%
6512     }%
6513   }%
6514   \SetGenericNewAcronym
6515   \GlsUseAcrStyleDefs{#1}%
6516   \@for\@gls@type:=\@glsacronymlists\do{%
6517     \defglsentryfmt[\@gls@type]{\GlsUseAcrEntryDispStyle{#1}}%
6518   }%
6519 }%
6520 }

```

`\newacronymstyle` `\newacronymstyle{<style name>}{<entry format definition>}{<display definitions>}`

Defines a new acronym style called *<style name>*.

```

6521 \newcommand*{\newacronymstyle}[3]{%
6522   \ifcsdef{@glsacr@dispstyle@#1}%
6523   {%
6524     \PackageError{glossaries}{Acronym style ‘#1’ already exists}{}%
6525   }%
6526   {%
6527     \csdef{@glsacr@dispstyle@#1}{#2}%
6528     \csdef{@glsacr@styledefs@#1}{#3}%
6529   }%
6530 }

```

`\renewacronymstyle` Redefines the given acronym style.

```

6531 \newcommand*{\renewacronymstyle}[3]{%
6532   \ifcsdef{@glsacr@dispstyle@#1}%
6533   {%
6534     \csdef{@glsacr@dispstyle@#1}{#2}%
6535     \csdef{@glsacr@styledefs@#1}{#3}%
6536   }%
6537   {%
6538     \PackageError{glossaries}{Acronym style ‘#1’ doesn’t exist}{}%
6539   }%
6540 }

```

rEntryDispStyle

```
6541 \newcommand*{\GlsUseAcrEntryDispStyle}[1]{\csuse{@glsacr@dispstyle@#1}}
```

UseAcrStyleDefs

```
6542 \newcommand*{\GlsUseAcrStyleDefs}[1]{\csuse{@glsacr@styledefs@#1}}
```

Predefined acronym styles:

long-short *<long>* (*<short>*) acronym style.

```
6543 \newacronymstyle{long-short}%
```

```
6544 {%
```

Check for long form in case this is a mixed glossary.

```
6545 \ifglshaslong{\glslabel}{\glsngenacfmt}{\glsngenentryfmt}%
```

```
6546 }%
```

```
6547 {%
```

```
6548 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
```

```
6549 \renewcommand*{\genacrfullformat}[2]{%
```

```
6550 \glentrylong{##1}##2\space
```

```
6551 (\protect\firstacronymfont{\glentryshort{##1}})%
```

```
6552 }%
```

```
6553 \renewcommand*{\Genacrfullformat}[2]{%
```

```
6554 \Glsentrylong{##1}##2\space
```

```
6555 (\protect\firstacronymfont{\glentryshort{##1}})%
```

```
6556 }%
```

```
6557 \renewcommand*{\genplacrfullformat}[2]{%
```

```
6558 \glentrylongpl{##1}##2\space
```

```
6559 (\protect\firstacronymfont{\glentryshortpl{##1}})%
```

```
6560 }%
```

```
6561 \renewcommand*{\Genplacrfullformat}[2]{%
```

```
6562 \Glsentrylongpl{##1}##2\space
```

```
6563 (\protect\firstacronymfont{\glentryshortpl{##1}})%
```

```
6564 }%
```

```
6565 \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
```

```
6566 \renewcommand*{\acronymsort}[2]{##1}%
```

```
6567 \renewcommand*{\acronymfont}[1]{##1}%
```

```
6568 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
```

```
6569 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
```

```
6570 }
```

long-sp-short Similar to the previous style but allows the space between the long and short form to be customized.

```
6571 \newacronymstyle{long-sp-short}%
```

```
6572 {%
```

Check for long form in case this is a mixed glossary.

```
6573 \ifglshaslong{\glslabel}{\glsngenacfmt}{\glsngenentryfmt}%
```

```
6574 }%
```

```
6575 {%
```

```
6576 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
```

```

6577 \renewcommand*{\genacrfullformat}[2]{%
6578 \glentrylong{##1}##2\glsacspace{##1}%
6579 (\protect\firstacronymfont{\glentryshort{##1}})%
6580 }%
6581 \renewcommand*{\Genacrfullformat}[2]{%
6582 \Glsentrylong{##1}##2\glsacspace{##1}%
6583 (\protect\firstacronymfont{\glentryshort{##1}})%
6584 }%
6585 \renewcommand*{\genplacrfullformat}[2]{%
6586 \glentrylongpl{##1}##2\glsacspace{##1}%
6587 (\protect\firstacronymfont{\glentryshortpl{##1}})%
6588 }%
6589 \renewcommand*{\Genplacrfullformat}[2]{%
6590 \Glsentrylongpl{##1}##2\glsacspace{##1}%
6591 (\protect\firstacronymfont{\glentryshortpl{##1}})%
6592 }%
6593 \renewcommand*{\acronymentry}[1]{\acronymfont{\glentryshort{##1}}}%
6594 \renewcommand*{\acronymsort}[2]{##1}%
6595 \renewcommand*{\acronymfont}[1]{##1}%
6596 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6597 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6598 }

```

`\glsacspace` Space between long and short form for the above style. This uses a non-breakable space if the short form is less than 3em, otherwise it uses a regular space.

```

6599 \newcommand*{\glsacspace}[1]{%
6600 \settowidth{\dimen0}{(\firstacronymfont{\glentryshort{##1}})}%
6601 \ifdim\dimen0<3em~\else\space\fi
6602 }

```

`short-long` (*short*) (*long*) acronym style.

```

6603 \newacronymstyle{short-long}%
6604 {%
    Check for long form in case this is a mixed glossary.
6605 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
6606 }%
6607 {%
6608 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6609 \renewcommand*{\genacrfullformat}[2]{%
6610 \protect\firstacronymfont{\glentryshort{##1}}##2\space
6611 (\glentrylong{##1})%
6612 }%
6613 \renewcommand*{\Genacrfullformat}[2]{%
6614 \protect\firstacronymfont{\Glsentryshort{##1}}##2\space
6615 (\glentrylong{##1})%
6616 }%
6617 \renewcommand*{\genplacrfullformat}[2]{%
6618 \protect\firstacronymfont{\glentryshortpl{##1}}##2\space

```

```

6619 (\glsentrylongpl{##1})%
6620 }%
6621 \renewcommand*{\Genplacrfullformat}[2]{%
6622 \protect\firstacronymfont{\Glsentryshortpl{##1}}##2\space
6623 (\glsentrylongpl{##1})%
6624 }%

6625 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6626 \renewcommand*{\acronymsort}[2]{##1}%
6627 \renewcommand*{\acronymfont}[1]{##1}%
6628 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
6629 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
6630 }

```

long-sc-short *<long>* (\textsc{<short>}) acronym style.

```

6631 \newacronymstyle{long-sc-short}%
6632 {%
6633 \GlsUseAcrEntryDisplayStyle{long-short}%
6634 }%
6635 {%
6636 \GlsUseAcrStyleDefs{long-short}%
6637 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6638 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6639 }

```

long-sm-short *<long>* (\textsmaller{<short>}) acronym style.

```

6640 \newacronymstyle{long-sm-short}%
6641 {%
6642 \GlsUseAcrEntryDisplayStyle{long-short}%
6643 }%
6644 {%
6645 \GlsUseAcrStyleDefs{long-short}%
6646 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6647 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6648 }

```

sc-short-long *<short>* (\textsc{<long>}) acronym style.

```

6649 \newacronymstyle{sc-short-long}%
6650 {%
6651 \GlsUseAcrEntryDisplayStyle{short-long}%
6652 }%
6653 {%
6654 \GlsUseAcrStyleDefs{short-long}%
6655 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
6656 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6657 }

```

sm-short-long *<short>* (\textsmaller{<long>}) acronym style.

```

6658 \newacronymstyle{sm-short-long}%

```

```

6659 {%
6660   \GlsUseAcrEntryDisplayStyle{short-long}%
6661 }%
6662 {%
6663   \GlsUseAcrStyleDefs{short-long}%
6664   \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6665   \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6666 }

```

long-short-desc *<long>* (*{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

6667 \newacronymstyle{long-short-desc}%
6668 {%
6669   \GlsUseAcrEntryDisplayStyle{long-short}%
6670 }%
6671 {%
6672   \GlsUseAcrStyleDefs{long-short}%
6673   \renewcommand*{\GenericAcronymFields}{}%
6674   \renewcommand*{\acronymsort}[2]{##2}%
6675   \renewcommand*{\acronymentry}[1]{%
6676     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6677 }

```

g-sp-short-desc *<long>* (*{<short>}*) acronym style that has an accompanying description (which the user needs to supply). The space between the long and short form is given by \glsacspace.

```

6678 \newacronymstyle{long-sp-short-desc}%
6679 {%
6680   \GlsUseAcrEntryDisplayStyle{long-sp-short}%
6681 }%
6682 {%
6683   \GlsUseAcrStyleDefs{long-sp-short}%
6684   \renewcommand*{\GenericAcronymFields}{}%
6685   \renewcommand*{\acronymsort}[2]{##2}%
6686   \renewcommand*{\acronymentry}[1]{%
6687     \glsentrylong{##1}\glsacspace{##1}(\acronymfont{\glsentryshort{##1}})}%
6688 }

```

g-sc-short-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

6689 \newacronymstyle{long-sc-short-desc}%
6690 {%
6691   \GlsUseAcrEntryDisplayStyle{long-sc-short}%
6692 }%
6693 {%
6694   \GlsUseAcrStyleDefs{long-sc-short}%
6695   \renewcommand*{\GenericAcronymFields}{}%
6696   \renewcommand*{\acronymsort}[2]{##2}%
6697   \renewcommand*{\acronymentry}[1]{%
6698     \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%

```

6699 }

g-sm-short-desc *⟨long⟩* (\textsmaller{⟨*short⟩*}) acronym style that has an accompanying description (which the user needs to supply).

```
6700 \newacronymstyle{long-sm-short-desc}%
6701 {%
6702   \GlsUseAcrEntryDispStyle{long-sm-short}%
6703 }%
6704 {%
6705   \GlsUseAcrStyleDefs{long-sm-short}%
6706   \renewcommand*{\GenericAcronymFields}{}%
6707   \renewcommand*{\acronymsort}[2]{##2}%
6708   \renewcommand*{\acronymentry}[1]{%
6709     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6710 }
```

short-long-desc *⟨short⟩* ({⟨*long⟩*}) acronym style that has an accompanying description (which the user needs to supply).

```
6711 \newacronymstyle{short-long-desc}%
6712 {%
6713   \GlsUseAcrEntryDispStyle{short-long}%
6714 }%
6715 {%
6716   \GlsUseAcrStyleDefs{short-long}%
6717   \renewcommand*{\GenericAcronymFields}{}%
6718   \renewcommand*{\acronymsort}[2]{##2}%
6719   \renewcommand*{\acronymentry}[1]{%
6720     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6721 }
```

short-long-desc *⟨long⟩* (\textsc{⟨*short⟩*}) acronym style that has an accompanying description (which the user needs to supply).

```
6722 \newacronymstyle{sc-short-long-desc}%
6723 {%
6724   \GlsUseAcrEntryDispStyle{sc-short-long}%
6725 }%
6726 {%
6727   \GlsUseAcrStyleDefs{sc-short-long}%
6728   \renewcommand*{\GenericAcronymFields}{}%
6729   \renewcommand*{\acronymsort}[2]{##2}%
6730   \renewcommand*{\acronymentry}[1]{%
6731     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6732 }
```

short-long-desc *⟨long⟩* (\textsmaller{⟨*short⟩*}) acronym style that has an accompanying description (which the user needs to supply).

```
6733 \newacronymstyle{sm-short-long-desc}%
6734 {%
```

```

6735 \GlsUseAcrEntryDispStyle{sm-short-long}%
6736 }%
6737 {%
6738 \GlsUseAcrStyleDefs{sm-short-long}%
6739 \renewcommand*{\GenericAcronymFields}{}%
6740 \renewcommand*{\acronymsort}[2]{##2}%
6741 \renewcommand*{\acronymentry}[1]{%
6742     \glentrylong{##1}\space (\acronymfont{\glentryshort{##1}})}%
6743 }

```

dua *<long>* only acronym style.

```

6744 \newacronymstyle{dua}%
6745 {%

```

Check for long form in case this is a mixed glossary.

```

6746 \ifdefempty\glscustomtext
6747 {%
6748     \ifglshaslong{\glslabel}%
6749     {%
6750         \glssifplural
6751         {%

```

Plural form:

```

6752         \glscapscase
6753         {%

```

Plural form, don't adjust case:

```

6754         \glentrylongpl{\glslabel}\glinsert
6755         }%
6756         {%

```

Plural form, make first letter upper case:

```

6757         \Glsentrylongpl{\glslabel}\glinsert
6758         }%
6759         {%

```

Plural form, all caps:

```

6760         \mfirstucMakeUppercase
6761         {\glentrylongpl{\glslabel}\glinsert}%
6762         }%
6763         }%
6764         {%

```

Singular form

```

6765         \glscapscase
6766         {%

```

Singular form, don't adjust case:

```

6767         \glentrylong{\glslabel}\glinsert
6768         }%
6769         {%

```

Subsequent singular form, make first letter upper case:

```
6770      \Glsentrylong{\glslabel}\glsinsert
6771      }%
6772      {%
```

Subsequent singular form, all caps:

```
6773      \mfirstucMakeUppercase
6774      {\glsentrylong{\glslabel}\glsinsert}%
6775      }%
6776      }%
6777      }%
6778      {%
```

Not an acronym:

```
6779      \glsgenentryfmt
6780      }%
6781      }%
6782      {\glscustomtext\glsinsert}%
6783      }%
6784      {%
6785      \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

6786      \renewcommand*{\acrfullfmt}[3]{%
6787      \glslink[##1]{##2}{\glsentrylong{##2}##3\space
6788      (\acronymfont{\glsentryshort{##2}})}}%
6789      \renewcommand*{\Acrfullfmt}[3]{%
6790      \glslink[##1]{##2}{\Glsentrylong{##2}##3\space
6791      (\acronymfont{\glsentryshort{##2}})}}%
6792      \renewcommand*{\ACRfullfmt}[3]{%
6793      \glslink[##1]{##2}{%
6794      \mfirstucMakeUppercase{\glsentrylong{##2}##3\space
6795      (\acronymfont{\glsentryshort{##2}})}}}%

6796      \renewcommand*{\acrfullplfmt}[3]{%
6797      \glslink[##1]{##2}{\glsentrylongpl{##2}##3\space
6798      (\acronymfont{\glsentryshortpl{##2}})}}%

6799      \renewcommand*{\Acrfullplfmt}[3]{%
6800      \glslink[##1]{##2}{\Glsentrylongpl{##2}##3\space
6801      (\acronymfont{\glsentryshortpl{##2}})}}%
6802      \renewcommand*{\ACRfullplfmt}[3]{%
6803      \glslink[##1]{##2}{%
6804      \mfirstucMakeUppercase{\glsentrylongpl{##2}##3\space
6805      (\acronymfont{\glsentryshortpl{##2}})}}}%
6806      \renewcommand*{\glsentryfull}[1]{%
6807      \glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
6808      }%
6809      \renewcommand*{\Glsentryfull}[1]{%
6810      \Glsentrylong{##1}\space(\acronymfont{\glsentryshort{##1}})%
6811      }%
```



```

6812 \renewcommand*{\glsentryfullpl}[1]{%
6813   \glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
6814 }%
6815 \renewcommand*{\Glsentryfullpl}[1]{%
6816   \Glsentrylongpl{##1}\space(\acronymfont{\glsentryshortpl{##1}})%
6817 }%
6818 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6819 \renewcommand*{\acronymsort}[2]{##1}%
6820 \renewcommand*{\acronymfont}[1]{##1}%
6821 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6822 }

```

dua-desc *<long>* only acronym style with user-supplied description.

```

6823 \newacronymstyle{dua-desc}%
6824 {%
6825   \GlsUseAcrEntryDispStyle{dua}%
6826 }%
6827 {%
6828   \GlsUseAcrStyleDefs{dua}%
6829   \renewcommand*{\GenericAcronymFields}{}%
6830   \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentrylong{##1}}}%
6831   \renewcommand*{\acronymsort}[2]{##2}%
6832 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```

6833 \newacronymstyle{footnote}%
6834 {%
6835   Check for long form in case this is a mixed glossary.
6836   \ifglshaslong{\glslabel}{\glsngenacfmt}{\glsngenentryfmt}%
6837 }%
6838 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
6839   Need to ensure hyperlinks are switched off on first use:
6840   \glshyperfirstfalse
6841   \renewcommand*{\genacrfullformat}[2]{%
6842     \protect\firstacronymfont{\glsentryshort{##1}}##2%
6843   }%
6844   \renewcommand*{\Genacrfullformat}[2]{%
6845     \firstacronymfont{\Glsentryshort{##1}}##2%
6846     \protect\footnote{\glsentrylong{##1}}%
6847   }%
6848   \renewcommand*{\genplacrfullformat}[2]{%
6849     \protect\firstacronymfont{\glsentryshortpl{##1}}##2%
6850     \protect\footnote{\glsentrylongpl{##1}}%
6851   }%
6852   \renewcommand*{\Genplacrfullformat}[2]{%

```

```

6853 \protect\firstacronymfont{\Glsentryshortpl{##1}}##2%
6854 \protect\footnote{\glsentrylongpl{##1}}%
6855 }%
6856 \renewcommand*{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}%
6857 \renewcommand*{\acronymsort}[2]{##1}%
6858 \renewcommand*{\acronymfont}[1]{##1}%
6859 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%

```

Don't use footnotes for \acrfull:

```

6860 \renewcommand*{\acrfullfmt}[3]{%
6861 \glslink[##1]{##2}{\acronymfont{\glsentryshort{##2}}##3\space
6862 (\glsentrylong{##2})}%
6863 \renewcommand*{\Acrfullfmt}[3]{%
6864 \glslink[##1]{##2}{\acronymfont{\Glsentryshort{##2}}##3\space
6865 (\glsentrylong{##2})}%
6866 \renewcommand*{\ACRfullfmt}[3]{%
6867 \glslink[##1]{##2}{%
6868 \mfirstucMakeUppercase{\acronymfont{\glsentryshort{##2}}##3\space
6869 (\glsentrylong{##2})}}}%
6870 \renewcommand*{\acrfullplfmt}[3]{%
6871 \glslink[##1]{##2}{\acronymfont{\glsentryshortpl{##2}}##3\space
6872 (\glsentrylongpl{##2})}%
6873 \renewcommand*{\Acrfullplfmt}[3]{%
6874 \glslink[##1]{##2}{\acronymfont{\Glsentryshortpl{##2}}##3\space
6875 (\glsentrylongpl{##2})}%
6876 \renewcommand*{\ACRfullplfmt}[3]{%
6877 \glslink[##1]{##2}{%
6878 \mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{##2}}##3\space
6879 (\glsentrylongpl{##2})}}}%

```

Similarly for \glsentryfull etc:

```

6880 \renewcommand*{\glsentryfull}[1]{%
6881 \acronymfont{\glsentryshort{##1}}\space(\glsentrylong{##1})}%
6882 \renewcommand*{\Glsentryfull}[1]{%
6883 \acronymfont{\Glsentryshort{##1}}\space(\glsentrylong{##1})}%
6884 \renewcommand*{\glsentryfullpl}[1]{%
6885 \acronymfont{\glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}%
6886 \renewcommand*{\Glsentryfullpl}[1]{%
6887 \acronymfont{\Glsentryshortpl{##1}}\space(\glsentrylongpl{##1})}%
6888 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

6889 \newacronymstyle{footnote-sc}%
6890 {%
6891 \GlsUseAcrEntryDispStyle{footnote}%
6892 }%
6893 {%
6894 \GlsUseAcrStyleDefs{footnote}%
6895 \renewcommand{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
6896 \renewcommand{\acronymfont}[1]{\textsc{##1}}%

```

```

6897 \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
6898 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

6899 \newacronymstyle{footnote-sm}%
6900 {%
6901 \GlsUseAcrEntryDisplayStyle{footnote}%
6902 }%
6903 {%
6904 \GlsUseAcrStyleDefs{footnote}%
6905 \renewcommand{\acronymentry}[1]{\acronymfont{\glsentryshort{##1}}}
6906 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
6907 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
6908 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

6909 \newacronymstyle{footnote-desc}%
6910 {%
6911 \GlsUseAcrEntryDisplayStyle{footnote}%
6912 }%
6913 {%
6914 \GlsUseAcrStyleDefs{footnote}%
6915 \renewcommand*{\GenericAcronymFields}{}%
6916 \renewcommand*{\acronymsort}[2]{##2}%
6917 \renewcommand*{\acronymentry}[1]{%
6918 \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6919 }

```

ootnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

6920 \newacronymstyle{footnote-sc-desc}%
6921 {%
6922 \GlsUseAcrEntryDisplayStyle{footnote-sc}%
6923 }%
6924 {%
6925 \GlsUseAcrStyleDefs{footnote-sc}%
6926 \renewcommand*{\GenericAcronymFields}{}%
6927 \renewcommand*{\acronymsort}[2]{##2}%
6928 \renewcommand*{\acronymentry}[1]{%
6929 \glsentrylong{##1}\space (\acronymfont{\glsentryshort{##1}})}%
6930 }

```

ootnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

6931 \newacronymstyle{footnote-sm-desc}%
6932 {%
6933 \GlsUseAcrEntryDisplayStyle{footnote-sm}%

```

```

6934 }%
6935 {%
6936   \GlsUseAcrStyleDefs{footnote-sm}%
6937   \renewcommand*{\GenericAcronymFields}{}%
6938   \renewcommand*{\acronymsort}[2]{##2}%
6939   \renewcommand*{\acronymentry}[1]{%
6940     \glstrylong{##1}\space (\acronymfont{\glstryshort{##1}})}%
6941 }

```

AcronymSynonyms

```

6942 \newcommand*{\DefineAcronymSynonyms}{%

```

Short form

```
\acs
```

```

6943   \let\acs\acrshort

```

First letter uppercase short form

```
\Acs
```

```

6944   \let\Acs\Acrshort

```

Plural short form

```
\acsp
```

```

6945   \let\acsp\acrshortpl

```

First letter uppercase plural short form

```
\Acsp
```

```

6946   \let\Acsp\Acrshortpl

```

Long form

```
\acl
```

```

6947   \let\acl\aclong

```

Plural long form

```
\aclp
```

```

6948   \let\aclp\aclongpl

```

First letter upper case long form

```
\Acl
```

```

6949   \let\Acl\Aclong

```

First letter upper case plural long form

```
\Aclp
```

```

6950   \let\Aclp\Aclongpl

```

Full form

`\acf`

6951 `\let\acf\acrfull`

Plural full form

`\acfp`

6952 `\let\acfp\acrfullpl`

First letter upper case full form

`\Acf`

6953 `\let\Acf\Acrfull`

First letter upper case plural full form

`\Acfp`

6954 `\let\Acfp\Acrfullpl`

Standard form

`\ac`

6955 `\let\ac\gls`

First upper case standard form

`\Ac`

6956 `\let\Ac\Gls`

Standard plural form

`\acp`

6957 `\let\acp\glspl`

Standard first letter upper case plural form

`\Acp`

6958 `\let\Acp\Glspl`

6959 }

Define synonyms if required

6960 `\ifglsacrshortcuts`

6961 `\DefineAcronymSynonyms`

6962 `\fi`

These commands for setting the style are now deprecated but are kept for backward compatibility.

`\AcronymDisplayStyle` Sets the default acronym display style for given glossary.

6963 `\newcommand*{\SetDefaultAcronymDisplayStyle}[1]{%`

6964 `\defglsentryfmt[#1]{\glsentryfmt}%`

6965 }

`\letNewAcronymDef` Sets up the acronym definition for the default style. The information is provided by the tokens `\glslabeltok`, `\glsshorttok`, `\glslongtok` and `\glskeylisttok`.

```

6966 \newcommand*{\DefaultNewAcronymDef}{%
6967   \edef\@do@newglossaryentry{%
6968     \noexpand\newglossaryentry{\the\glslabeltok}%
6969     {%
6970       type=\acronymtype,%
6971       name={\the\glsshorttok},%
6972       sort={\the\glsshorttok},%
6973       text={\the\glsshorttok},%
6974       first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
6975       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
6976       firstplural={\acrfullformat{\noexpand\expandonce\noexpand\@glo@longpl}%
6977                   {\noexpand\expandonce\noexpand\@glo@shortpl}},%
6978       short={\the\glsshorttok},%
6979       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
6980       long={\the\glslongtok},%
6981       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
6982       description={\the\glslongtok},%
6983       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%

```

Remaining options specified by the user:

```

6984     \the\glskeylisttok
6985   }%
6986 }%
6987 \let\@org@gls@assign@firstpl\gls@assign@firstpl
6988 \let\@org@gls@assign@plural\gls@assign@plural
6989 \let\@org@gls@assign@descplural\gls@assign@descplural
6990 \def\gls@assign@firstpl##1##2{%
6991   \@gls@expand@field{##1}{firstpl}{##2}%
6992 }%
6993 \def\gls@assign@plural##1##2{%
6994   \@gls@expand@field{##1}{plural}{##2}%
6995 }%
6996 \def\gls@assign@descplural##1##2{%
6997   \@gls@expand@field{##1}{descplural}{##2}%
6998 }%
6999 \@do@newglossaryentry
7000 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7001 \let\gls@assign@plural\@org@gls@assign@plural
7002 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7003 }

```

`\letAcronymStyle` Set up the default acronym style:

```

7004 \newcommand*{\SetDefaultAcronymStyle}{%
    Set the display style:
7005   \@for\@gls@type:=\@glsacronymlists\do{%
7006     \SetDefaultAcronymDisplayStyle{\@gls@type}%
7007   }%

```

Set up the definition of `\newacronym`:

```
7008 \renewcommand{\newacronym}[4][]{%
```

If user is just using the main glossary and hasn't identified it as a list of acronyms, then update.

(This is done to ensure backwards compatibility with versions prior to 2.04).

```
7009 \ifx\@glsacronymlists\@empty
7010 \def\@glo@type{\acronymtype}%
7011 \setkeys{glossentry}{##1}%
7012 \DeclareAcronymList{\@glo@type}%
7013 \SetDefaultAcronymDisplayStyle{\@glo@type}%
7014 \fi
7015 \glskeylisttok{##1}%
7016 \glslabeltok{##2}%
7017 \glsshorttok{##3}%
7018 \gslongtok{##4}%
7019 \newacronymhook
7020 \DefaultNewAcronymDef
7021 }%
7022 \renewcommand*{\acrpluralsuffix}{\glsacrpluralsuffix}%
7023 }
```

`\acrfootnote` Used by the footnote acronym styles.

```
7024 \newcommand*{\acrfootnote}[3]{\acrlinkfootnote{#1}{#2}{#3}}
```

`acrlinkfootnote`

```
7025 \newcommand*{\acrlinkfootnote}[3]{%
7026 \footnote{\glslink{#1}{#2}{#3}}%
7027 }
```

`acrnolinkfootnote`

```
7028 \newcommand*{\acrnolinkfootnote}[3]{%
7029 \footnote{#3}%
7030 }
```

`\acronymDisplayStyle` Sets the acronym display style for given glossary for the description and footnote combination.

```
7031 \newcommand*{\SetDescriptionFootnoteAcronymDisplayStyle}[1]{%
7032 \defglsentryfmt{#1}{%
7033 \ifdefempty\glscustomtext
7034 {%
7035 \ifglsused{\glslabel}%
7036 {%
7037 \acronymfont{\glsgenentryfmt}%
7038 }%
7039 {%
7040 \firstacronymfont{\glsgenentryfmt}%
7041 \ifglsymbol{\glslabel}%
7042 {%
```

```

7043         \expandafter\protect\expandafter\acrfootnote\expandafter
7044         {\@gls@link@opts}{\@gls@link@label}%
7045         {%
7046         \glsifplural
7047         {\glsentrysymbolplural{\glslabel}}%
7048         {\glsentrysymbol{\glslabel}}%
7049         }%
7050     }%
7051 }%
7052 }%
7053 {\glscustomtext\glsinsert}%
7054 }%
7055 }

```

teNewAcronymDef

```

7056 \newcommand*{\DescriptionFootnoteNewAcronymDef}{%
7057 \edef\@do@newglossaryentry{%
7058 \noexpand\newglossaryentry{\the\glslabeltok}%
7059 {%
7060     type=\acronymtype,%
7061     name={\noexpand\acronymfont{\the\glsshorttok}},%
7062     sort={\the\glsshorttok},%
7063     first={\the\glsshorttok},%
7064     firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7065     text={\the\glsshorttok},%
7066     plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7067     short={\the\glsshorttok},%
7068     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7069     long={\the\glslongtok},%
7070     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7071     symbol={\the\glslongtok},%
7072     symbolplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7073     \the\glskeylisttok
7074 }%
7075 }%
7076 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7077 \let\@org@gls@assign@plural\gls@assign@plural
7078 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7079 \def\gls@assign@firstpl##1##2{%
7080     \@gls@expand@field{##1}{firstpl}{##2}%
7081 }%
7082 \def\gls@assign@plural##1##2{%
7083     \@gls@expand@field{##1}{plural}{##2}%
7084 }%
7085 \def\gls@assign@symbolplural##1##2{%
7086     \@gls@expand@field{##1}{symbolplural}{##2}%
7087 }%
7088 \@do@newglossaryentry
7089 \let\gls@assign@plural\@org@gls@assign@plural

```



```

7090 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7091 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7092 }

```

oteAcronymStyle If a description and footnote are both required, store the long form in the symbol key. Store the short form in text key. Note that since the long form is stored in the symbol key, if you want the long form to appear in the list of acronyms, you need to use a glossary style that displays the symbol key.

```

7093 \newcommand*{\SetDescriptionFootnoteAcronymStyle}{%
7094   \renewcommand{\newacronym}[4][\]{%
7095     \ifx\@glsacronymlists\@empty
7096       \def\@glo@type{\acronymtype}%
7097       \setkeys{glossentry}{##1}%
7098       \DeclareAcronymList{\@glo@type}%
7099       \SetDescriptionFootnoteAcronymDisplayStyle{\@glo@type}%
7100     \fi
7101     \glskeylisttok{##1}%
7102     \glslabeltok{##2}%
7103     \glsshorttok{##3}%
7104     \glslongtok{##4}%
7105     \newacronymhook
7106     \DescriptionFootnoteNewAcronymDef
7107   }%

```

If footnote package option is specified, set the first use to append the long form (stored in symbol) as a footnote.

```

7108 \@for\@gls@type:=\@glsacronymlists\do{%
7109   \SetDescriptionFootnoteAcronymDisplayStyle{\@gls@type}%
7110 }%

```

Redefine `\acronymfont` if small caps required. The plural suffix is set in an upright font so that it remains in normal lower case, otherwise it looks as though it's part of the acronym.

```

7111 \ifglsacrsmallcaps
7112   \renewcommand*{\acronymfont}[1]{\textsc{##1}}%
7113   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7114 \else
7115   \ifglsacrsmaller
7116     \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7117   \fi
7118 \fi

```

Check for package option clash

```

7119 \ifglsacrdua
7120   \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
7121     can’t both be set}{}%
7122 \fi
7123 }%

```

nymDisplayStyle Sets the acronym display style for given glossary with description and dua combination.

```

7124 \newcommand*{\SetDescriptionDUAAcronymDisplayStyle}[1]{%
7125   \def\glsentryfmt[#1]{\glsentryfmt}%
7126 }

```

UANewAcronymDef

```

7127 \newcommand*{\DescriptionDUANewAcronymDef}{%
7128   \edef\@do@newglossaryentry{%
7129     \noexpand\newglossaryentry{\the\glslabeltok}%
7130     {%
7131       type=\acronymtype,%
7132       name={\the\glslongtok},%
7133       sort={\the\glslongtok},%
7134       text={\the\glslongtok},%
7135       first={\the\glslongtok},%
7136       plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7137       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7138       short={\the\glsshorttok},%
7139       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7140       long={\the\glslongtok},%
7141       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7142       symbol={\the\glsshorttok},%
7143       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7144       \the\glskeylisttok
7145     }%
7146   }%
7147   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7148   \let\@org@gls@assign@plural\gls@assign@plural
7149   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7150   \def\gls@assign@firstpl##1##2{%
7151     \@@gls@expand@field{##1}{firstpl}{##2}%
7152   }%
7153   \def\gls@assign@plural##1##2{%
7154     \@@gls@expand@field{##1}{plural}{##2}%
7155   }%
7156   \def\gls@assign@symbolplural##1##2{%
7157     \@@gls@expand@field{##1}{symbolplural}{##2}%
7158   }%
7159   \@do@newglossaryentry
7160   \let\gls@assign@firstpl\@org@gls@assign@firstpl
7161   \let\gls@assign@plural\@org@gls@assign@plural
7162   \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7163 }

```

DUAAcronymStyle Description, don't use acronym and no footnote. Note that the short form is stored in the symbol key, so if the short form needs to be displayed in the glossary, use a style the displays the symbol.

```

7164 \newcommand*{\SetDescriptionDUAAcronymStyle}{%
7165   \ifglssacrsmallcaps
7166     \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'

```

```

7167     can't both be set}{}%
7168 \else
7169     \ifglacrsmaller
7170         \PackageError{glossaries}{Option clash: 'smaller' and 'dua'
7171             can't both be set}{}%
7172     \fi
7173 \fi
7174 \renewcommand{\newacronym}[4][]{%
7175     \ifx\@glsacronymlists\@empty
7176         \def\@glo@type{\acronymtype}%
7177         \setkeys{glossentry}{##1}%
7178         \DeclareAcronymList{\@glo@type}%
7179         \SetDescriptionDUAAcronymDisplayStyle{\@glo@type}%
7180     \fi
7181     \glskeylisttok{##1}%
7182     \glslabeltok{##2}%
7183     \glsshorttok{##3}%
7184     \glslongtok{##4}%
7185     \newacronymhook
7186     \DescriptionDUANewAcronymDef
7187 }%

Set display.
7188 \@for\@gls@type:=\@glsacronymlists\do{%
7189     \SetDescriptionDUAAcronymDisplayStyle{\@gls@type}%
7190 }%
7191 }%

```

`\glsAcronymDisplayStyle` Sets the acronym display style for given glossary using the description setting (but not footnote or dua).

```

7192 \newcommand*{\SetDescriptionAcronymDisplayStyle}[1]{%
7193     \defglentryfmt[##1]{%

7194         \ifdefempty\glscustomtext
7195         {%
7196             \ifglshassymbol{\glslabel}%
7197             {%

Move the inserted text outside of \acronymfont
7198                 \let\gls@org@insert\glsinsert
7199                 \let\glsinsert\@empty
7200                 \acronymfont{\glsgenentryfmt}\gls@org@insert
7201             }%
7202         {%
7203             \glsgenentryfmt
7204             \ifglshassymbol{\glslabel}%
7205             {%
7206                 \glsifplural
7207                 {%
7208                     \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%

```

```

7209         }%
7210         {%
7211         \def\@glo@symbol{\glsentrysymbol{\glslabel}}}%
7212         }%
7213         \space(\protect\firstacronymfont
7214         {\glscapscase
7215         {\@glo@symbol}
7216         {\@glo@symbol}
7217         {\mfirstucMakeUppercase{\@glo@symbol}}})}%
7218     }%
7219     {}%
7220 }%
7221 }%
7222 {\glscustomtext\glsinsert}%
7223 }%
7224 }

```

onNewAcronymDef

```

7225 \newcommand*{\DescriptionNewAcronymDef}{%
7226 \edef\@do@newglossaryentry{%
7227 \noexpand\newglossaryentry{\the\glslabeltok}%
7228 {%
7229 type=\acronymtype,%
7230 name={\noexpand
7231 \acronymformat{\the\glsshorttok}{\the\glslongtok}},%
7232 sort={\the\glsshorttok},%
7233 first={\the\glslongtok},%
7234 firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7235 text={\the\glsshorttok},%
7236 plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7237 short={\the\glsshorttok},%
7238 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7239 long={\the\glslongtok},%
7240 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7241 symbol={\noexpand\@glo@text},%
7242 symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7243 \the\glskeylisttok}%
7244 }%
7245 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7246 \let\@org@gls@assign@plural\gls@assign@plural
7247 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7248 \def\gls@assign@firstpl##1##2{%
7249 \@@gls@expand@field{##1}{firstpl}{##2}%
7250 }%
7251 \def\gls@assign@plural##1##2{%
7252 \@@gls@expand@field{##1}{plural}{##2}%
7253 }%
7254 \def\gls@assign@symbolplural##1##2{%
7255 \@@gls@expand@field{##1}{symbolplural}{##2}%

```

```

7256 }%
7257 \@do@newglossaryentry
7258 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7259 \let\gls@assign@plural\@org@gls@assign@plural
7260 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7261 }

```

ionAcronymStyle Option description is used, but not dua or footnote. Store long form in first key and short form in text and symbol key. The name is stored using `\acrnameformat` to allow the user to override the way the name is displayed in the list of acronyms.

```

7262 \newcommand*{\SetDescriptionAcronymStyle}{%
7263   \renewcommand{\newacronym}[4][\]{%
7264     \ifx\@glsacronymlists\@empty
7265       \def\@glo@type{\acronymtype}%
7266       \setkeys{glossentry}{##1}%
7267       \DeclareAcronymList{\@glo@type}%
7268       \SetDescriptionAcronymDisplayStyle{\@glo@type}%
7269     \fi
7270     \glskeylisttok{##1}%
7271     \glslabeltok{##2}%
7272     \glsshorttok{##3}%
7273     \glslongtok{##4}%
7274     \newacronymhook
7275     \DescriptionNewAcronymDef
7276   }%

```

Set display.

```

7277 \@for\@gls@type:=\@glsacronymlists\do{%
7278   \SetDescriptionAcronymDisplayStyle{\@gls@type}%
7279 }%

```

Redefine `\acronymfont` if small caps required. The plural suffix is set in an upright font so that it remains in normal lower case, otherwise it looks as though it's part of the acronym.

```

7280 \ifglsacrsmallcaps
7281   \renewcommand{\acronymfont}[1]{\textsc{##1}}
7282   \renewcommand*{\acrpluralsuffix}{\glsupacrpluralsuffix}%
7283 \else
7284   \ifglsacrsmaller
7285     \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7286   \fi
7287 \fi
7288 }%

```

nymDisplayStyle Sets the acronym display style for given glossary with footnote setting (but not description or dua).

```

7289 \newcommand*{\SetFootnoteAcronymDisplayStyle}[1]{%
7290   \defglsentryfmt[##1]{%
7291     \ifdefempty\glscustomtext
7292     {%

```

Move the inserted text outside of \acronymfont

```

7293 \let\gls@org@insert\glsinsert
7294 \let\glsinsert\@empty
7295 \ifglsused{\glslabel}%
7296 {%
7297   \acronymfont{\gls@genentryfmt}\gls@org@insert
7298 }%
7299 {%
7300   \firstacronymfont{\gls@genentryfmt}\gls@org@insert
7301   \ifgls@haslong{\glslabel}%
7302   {%
7303     \expandafter\protect\expandafter\acrfootnote\expandafter
7304     {\@gls@link@opts}{\@gls@link@label}%
7305     {%
7306       \glsifplural
7307         {\glsentrylongpl{\glslabel}}%
7308         {\glsentrylong{\glslabel}}%
7309     }%
7310   }%
7311   {}%
7312 }%
7313 }%
7314 {\gls@customtext\glsinsert}%
7315 }%
7316 }

```

teNewAcronymDef

```

7317 \newcommand*{\FootnoteNewAcronymDef}{%
7318   \edef\@do@newglossaryentry{%
7319     \noexpand\newglossaryentry{\the\glslabeltok}%
7320     {%
7321       type=\acronymtype,%
7322       name={\noexpand\acronymfont{\the\glsshorttok}},%
7323       sort={\the\glsshorttok},%
7324       text={\the\glsshorttok},%
7325       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7326       first={\the\glsshorttok},%
7327       firstplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7328       short={\the\glsshorttok},%
7329       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7330       long={\the\glslongtok},%
7331       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7332       description={\the\glslongtok},%
7333       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7334       \the\glskeylisttok
7335     }%
7336   }%
7337   \let\@org@gls@assign@plural\gls@assign@plural

```

```

7338 \let\@org@gl@s@assign@firstpl\gl@s@assign@firstpl
7339 \let\@org@gl@s@assign@descplural\gl@s@assign@descplural
7340 \def\gl@s@assign@firstpl##1##2{%
7341   \@@gl@s@expand@field{##1}{firstpl}{##2}%
7342 }%
7343 \def\gl@s@assign@plural##1##2{%
7344   \@@gl@s@expand@field{##1}{plural}{##2}%
7345 }%
7346 \def\gl@s@assign@descplural##1##2{%
7347   \@@gl@s@expand@field{##1}{descplural}{##2}%
7348 }%
7349 \do@newglossaryentry
7350 \let\gl@s@assign@plural\@org@gl@s@assign@plural
7351 \let\gl@s@assign@firstpl\@org@gl@s@assign@firstpl
7352 \let\gl@s@assign@descplural\@org@gl@s@assign@descplural
7353 }

```

oteAcronymStyle If footnote package option is specified, set the first use to append the long form (stored in description) as a footnote. Use the description key to store the long form.

```

7354 \newcommand*{\SetFootnoteAcronymStyle}{%
7355   \renewcommand{\newacronym}[4][]{%
7356     \ifx\@gl@sacronymlists\@empty
7357       \def\@glo@type{\acronymtype}%
7358       \setkeys{glossentry}{##1}%
7359       \DeclareAcronymList{\@glo@type}%
7360       \SetFootnoteAcronymDisplayStyle{\@glo@type}%
7361     \fi
7362     \gl@keylisttok{##1}%
7363     \gl@labeltok{##2}%
7364     \gl@shorttok{##3}%
7365     \gl@longtok{##4}%
7366     \newacronymhook
7367     \FootnoteNewAcronymDef
7368   }%

```

Set display

```

7369 \@for\@gl@s@type:=\@gl@sacronymlists\do{%
7370   \SetFootnoteAcronymDisplayStyle{\@gl@s@type}%
7371 }%

```

Redefine \acronymfont if small caps required. The plural suffix is set in an upright font so that it remains in normal lower case, otherwise it looks as though it's part of the acronym.

```

7372 \ifgl@sacrsmallcaps
7373   \renewcommand*{\acronymfont}[1]{\textsc{##1}}%
7374   \renewcommand*{\acrpluralsuffix}{\gl@supacrpluralsuffix}%
7375 \else
7376   \ifgl@sacrsmaller
7377     \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
7378   \fi
7379 \fi

```

Check for option clash

```
7380 \ifglsacrdue
7381 \PackageError{glossaries}{Option clash: 'footnote' and 'dua'
7382 can't both be set}}{}%
7383 \fi
7384 }%
```

`\parenifnotempty` Do a space followed by the argument if the argument doesn't expand to empty or `\relax`. If argument isn't empty (or `\relax`), apply the macro to it given in the second argument.

```
7385 \DeclareRobustCommand*\glsdoparenifnotempty}[2]{%
7386 \protected@edef\gls@tmp{#1}%
7387 \ifdefempty\gls@tmp
7388 {}%
7389 {%
7390 \ifx\gls@tmp\@gls@default@value
7391 \else
7392 \space (#2{#1})%
7393 \fi
7394 }%
7395 }
```

`\nymDisplayStyle` Sets the acronym display style for given glossary where neither footnote nor description is required, but smallcaps or smaller specified.

```
7396 \newcommand*\SetSmallAcronymDisplayStyle}[1]{%
7397 \defglsentryfmt[#1]{%
7398 \ifdefempty\glscustomtext
7399 {%
```

Move the inserted text outside of `\acronymfont`

```
7400 \let\gls@org@insert\glsinsert
7401 \let\glsinsert\@empty
7402 \ifglsused{\glslabel}%
7403 {%
7404 \acronymfont{\glsgenentryfmt}\gls@org@insert
7405 }%
7406 {%
7407 \glsgenentryfmt
7408 \ifgls hassymbol{\glslabel}%
7409 {%
7410 \glsifplural
7411 {%
7412 \def\@glo@symbol{\glsentrysymbolplural{\glslabel}}%
7413 }%
7414 {%
7415 \def\@glo@symbol{\glsentrysymbol{\glslabel}}%
7416 }%
7417 \space
7418 (\glscaps case
```



```

7419         {\firstacronymfont{\@glo@symbol}}}%
7420         {\firstacronymfont{\@glo@symbol}}}%
7421         {\firstacronymfont{\mfirstucMakeUppercase{\@glo@symbol}}})}%
7422     }%
7423     {}%
7424 }%
7425 }%
7426 {\glscustomtext\glsinsert}%
7427 }%
7428 }

```

11NewAcronymDef

```

7429 \newcommand*{\SmallNewAcronymDef}{%
7430   \edef\@do@newglossaryentry{%
7431     \noexpand\newglossaryentry{\the\glslabeltok}%
7432     {%
7433       type=\acronymtype,%
7434       name={\noexpand\acronymfont{\the\glsshorttok}},%
7435       sort={\the\glsshorttok},%
7436       text={\the\glsshorttok},%
7437
7438       Default to the short plural.
7439       plural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7440       first={\the\glslongtok},%
7441
7442       Default to the long plural.
7443       firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7444       short={\the\glsshorttok},%
7445       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7446       long={\the\glslongtok},%
7447       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7448       description={\noexpand\@glo@first},%
7449       descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7450       symbol={\the\glsshorttok},%
7451
7452       Default to the short plural.
7453       symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7454       \the\glskeylisttok
7455     }%
7456   }%
7457   \let\@org@gls@assign@firstpl\gls@assign@firstpl
7458   \let\@org@gls@assign@plural\gls@assign@plural
7459   \let\@org@gls@assign@descplural\gls@assign@descplural
7460   \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7461   \def\gls@assign@firstpl##1##2{%
7462     \@gls@expand@field{##1}{firstpl}{##2}%
7463   }%
7464   \def\gls@assign@plural##1##2{%
7465     \@gls@expand@field{##1}{plural}{##2}%
7466   }%

```

```

7461 \def\gls@assign@descplural##1##2{%
7462   \@gls@expand@field{##1}{descplural}{##2}%
7463 }%
7464 \def\gls@assign@symbolplural##1##2{%
7465   \@gls@expand@field{##1}{symbolplural}{##2}%
7466 }%
7467 \do@newglossaryentry
7468 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7469 \let\gls@assign@plural\@org@gls@assign@plural
7470 \let\gls@assign@descplural\@org@gls@assign@descplural
7471 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7472 }

```

`allAcronymStyle` Neither footnote nor description required, but smallcaps or smaller specified. Use the symbol key to store the short form and first to store the long form.

```

7473 \newcommand*\SetSmallAcronymStyle{%
7474   \renewcommand{\newacronym}[4][]{%
7475     \ifx\glsacronymlists\@empty
7476       \def\@glo@type{\acronymtype}%
7477       \setkeys{glossentry}{##1}%
7478       \DeclareAcronymList{\@glo@type}%
7479       \SetSmallAcronymDisplayStyle{\@glo@type}%
7480     \fi
7481     \glskeylisttok{##1}%
7482     \glslabeltok{##2}%
7483     \glsshorttok{##3}%
7484     \glslongtok{##4}%
7485     \newacronymhook
7486     \SmallNewAcronymDef
7487   }%

```

Change the display since first only contains long form.

```

7488 \@for\@gls@type:=\@glsacronymlists\do{%
7489   \SetSmallAcronymDisplayStyle{\@gls@type}%
7490 }%

```

Redefine `\acronymfont` if small caps required. The plural suffix is set in an upright font so that it remains in normal lower case, otherwise it looks as though it's part of the acronym.

```

7491 \ifglsacrsmallcaps
7492   \renewcommand*\acronymfont[1]{\textsc{##1}}
7493   \renewcommand*\acrpluralsuffix{\glsupacrpluralsuffix}%
7494 \else
7495   \renewcommand*\acronymfont[1]{\textsmaller{##1}}
7496 \fi

```

check for option clash

```

7497 \ifglsacrdua
7498   \ifglsacrsmallcaps
7499     \PackageError{glossaries}{Option clash: ‘smallcaps’ and ‘dua’
7500       can’t both be set}{}%

```

```

7501 \else
7502 \PackageError{glossaries}{Option clash: ‘smaller’ and ‘dua’
7503 can’t both be set}{}%
7504 \fi
7505 \fi
7506 }%

```

DUADisplayStyle Sets the acronym display style for given glossary with dua setting.

```

7507 \newcommand*{\SetDUADisplayStyle}[1]{%
7508 \defglentryfmt[#1]{\glsgenentryfmt}%
7509 }

```

UANewAcronymDef

```

7510 \newcommand*{\DUANewAcronymDef}{%
7511 \edef\@do@newglossaryentry{%
7512 \noexpand\newglossaryentry{\the\glslabeltok}%
7513 {%
7514 type=\acronymtype,%
7515 name={\the\glsshorttok},%
7516 text={\the\glslongtok},%
7517 first={\the\glslongtok},%
7518 plural={\noexpand\expandonce\noexpand\@glo@longpl},%
7519 firstplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7520 short={\the\glsshorttok},%
7521 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7522 long={\the\glslongtok},%
7523 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7524 description={\the\glslongtok},%
7525 descriptionplural={\noexpand\expandonce\noexpand\@glo@longpl},%
7526 symbol={\the\glsshorttok},%
7527 symbolplural={\noexpand\expandonce\noexpand\@glo@shortpl},%
7528 \the\glskeylisttok
7529 }%
7530 }%
7531 \let\@org@gls@assign@firstpl\gls@assign@firstpl
7532 \let\@org@gls@assign@plural\gls@assign@plural
7533 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
7534 \let\@org@gls@assign@descplural\gls@assign@descplural
7535 \def\gls@assign@firstpl##1##2{%
7536 \@@gls@expand@field{##1}{firstpl}{##2}%
7537 }%
7538 \def\gls@assign@plural##1##2{%
7539 \@@gls@expand@field{##1}{plural}{##2}%
7540 }%
7541 \def\gls@assign@symbolplural##1##2{%
7542 \@@gls@expand@field{##1}{symbolplural}{##2}%
7543 }%
7544 \def\gls@assign@descplural##1##2{%
7545 \@@gls@expand@field{##1}{descplural}{##2}%

```

```

7546 }%
7547 \do@newglossaryentry
7548 \let\gls@assign@firstpl\@org@gls@assign@firstpl
7549 \let\gls@assign@plural\@org@gls@assign@plural
7550 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
7551 \let\gls@assign@descplural\@org@gls@assign@descplural
7552 }

```

`\SetDUASyle` Always expand acronyms.

```

7553 \newcommand*{\SetDUASyle}{%
7554   \renewcommand{\newacronym}[4][]{%
7555     \ifx\@glsacronymlists\empty
7556       \def\@glo@type{\acronymtype}%
7557       \setkeys{glossentry}{##1}%
7558       \DeclareAcronymList{\@glo@type}%
7559       \SetDUADisplayStyle{\@glo@type}%
7560     \fi
7561     \glskeylisttok{##1}%
7562     \glslabeltok{##2}%
7563     \glsshorttok{##3}%
7564     \glslongtok{##4}%
7565     \newacronymhook
7566     \DUANewAcronymDef
7567   }%
7568   \@for\@gls@type:=\@glsacronymlists\do{%
7569     \SetDUADisplayStyle{\@gls@type}%
7570   }%
7571 }

```

Set the display

`SetAcronymStyle`

```

7572 \newcommand*{\SetAcronymStyle}{%
7573   \SetDefaultAcronymStyle
7574   \ifglsacrdescription
7575     \ifglsacrfootnote
7576       \SetDescriptionFootnoteAcronymStyle
7577     \else
7578       \ifglsacrdua
7579         \SetDescriptionDUAAcronymStyle
7580       \else
7581         \SetDescriptionAcronymStyle
7582       \fi
7583     \fi
7584   \else
7585     \ifglsacrfootnote
7586       \SetFootnoteAcronymStyle
7587     \else
7588       \ifthenelse{\boolean{glsacrsmallicaps}}{OR
7589         \boolean{glsacrsmaller}}}%

```

```

7590      {%
7591      \SetSmallAcronymStyle
7592      }%
7593      {%
7594      \ifglssacrdua
7595      \SetDUASStyle
7596      \fi
7597      }%
7598      \fi
7599      \fi
7600 }

```

Set the acronym style according to the package options

```

7601 \SetAcronymStyle

```

Allow user to define their own custom acronyms. (For compatibility with versions before v3.0, the short form is stored in the user1 key, the plural short form is stored in the user2 key, the long form is stored in the user3 key and the plural long form is stored in the user4 key.) Defaults to displaying only the acronym with the long form as the description.

`\setacronymstyle` Sets the acronym display style.

```

7602 \newcommand*{\SetCustomDisplayStyle}[1]{%
7603   \defglssentryfmt[#1]{\glsgenentryfmt}%
7604 }

```

`\setacronymfields`

```

7605 \newcommand*{\CustomAcronymFields}{%
7606   name={\the\glsshorttok},%
7607   description={\the\glslongtok},%
7608   first={\acrfullformat{\the\glslongtok}{\the\glsshorttok}},%
7609   firstplural={\acrfullformat
7610     {\noexpand\glssentrylongpl{\the\glslabeltok}}}%
7611     {\noexpand\glssentryshortpl{\the\glslabeltok}}},%
7612   text={\the\glsshorttok},%
7613   plural={\the\glsshorttok\noexpand\acrpluralsuffix}%
7614 }

```

`\setnewacronymdef`

```

7615 \newcommand*{\CustomNewAcronymDef}{%
7616   \protected@edef\@do@newglossaryentry{%
7617     \noexpand\newglossaryentry{\the\glslabeltok}%
7618     {%
7619       type=\acronymtype,%
7620       short={\the\glsshorttok},%
7621       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
7622       long={\the\glslongtok},%
7623       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
7624       user1={\the\glsshorttok},%

```

```

7625     user2={\the\glsshorttok\noexpand\acrpluralsuffix},%
7626     user3={\the\glslongtok},%
7627     user4={\the\glslongtok\noexpand\acrpluralsuffix},%
7628     \CustomAcronymFields,%
7629     \the\glskeylisttok
7630 }%
7631 }%
7632 \@do@newglossaryentry
7633 }

```

`\SetCustomStyle`

```

7634 \newcommand*{\SetCustomStyle}{%
7635   \renewcommand{\newacronym}[4][]{%
7636     \ifx\@glsacronymlists\@empty
7637       \def\@glo@type{\acronymtype}%
7638       \setkeys{glossentry}{##1}%
7639       \DeclareAcronymList{\@glo@type}%
7640       \SetCustomDisplayStyle{\@glo@type}%
7641     \fi
7642     \glskeylisttok{##1}%
7643     \glslabeltok{##2}%
7644     \glsshorttok{##3}%
7645     \glslongtok{##4}%
7646     \newacronymhook
7647     \CustomNewAcronymDef
7648   }%
7649   \SetCustomDisplayStyle{\@glo@type}%
7650   \SetCustomDisplayStyle{\@gls@type}%
7651 }%
7652 }

```

1.19 Predefined Glossary Styles

The glossaries bundle comes with some predefined glossary styles. These need to be loaded now for the style option to use them.

First, the glossary hyper-navigation commands need to be loaded.

```
7653 \RequirePackage{glossary-hypernav}
```

The styles that use list-like environments. These are not loaded if the `nolist` option is used:

```
7654 \@gls@loadlist
```

The styles that use the `longtable` environment. These are not loaded if the `nolong` package option is used.

```
7655 \@gls@loadlong
```

The styles that use the `supertabular` environment. These are not loaded if the `nosuper` package option is used or if the package isn't installed.

```
7656 \@gls@loadsuper
```

The tree-like styles. These are not loaded if the notree package option is used.

```
7657 \@gls@loadtree
```

The default glossary style is set according to the style package option, but can be overridden by `\glossarystyle`. The required style must be defined at this point.

```
7658 \ifx\@glossary@default@style\relax
```

```
7659 \else
```

```
7660   \setglossarystyle{\@glossary@default@style}
```

```
7661 \fi
```

1.20 Debugging Commands

`\showgloparent` `\showgloparent{<label>}`

```
7662 \newcommand*{\showgloparent}[1]{%
```

```
7663   \expandafter\show\csname glo@\glsdetoklabel{#1}@parent\endcsname
```

```
7664 }
```

`\showglolevel` `\showglolevel{<label>}`

```
7665 \newcommand*{\showglolevel}[1]{%
```

```
7666   \expandafter\show\csname glo@\glsdetoklabel{#1}@level\endcsname
```

```
7667 }
```

`\showglotext` `\showglotext{<label>}`

```
7668 \newcommand*{\showglotext}[1]{%
```

```
7669   \expandafter\show\csname glo@\glsdetoklabel{#1}@text\endcsname
```

```
7670 }
```

`\showgloplural` `\showgloplural{<label>}`

```
7671 \newcommand*{\showgloplural}[1]{%
```

```
7672   \expandafter\show\csname glo@\glsdetoklabel{#1}@plural\endcsname
```

```
7673 }
```

`\showglofirst` `\showglofirst{<label>}`

```

7674 \newcommand*{\showglofirst}[1]{%
7675   \expandafter\show\csname glo\glsdetoklabel{#1}@first\endcsname
7676 }

```

\showglofirstpl \showglofirstpl{\label{}}

```

7677 \newcommand*{\showglofirstpl}[1]{%
7678   \expandafter\show\csname glo\glsdetoklabel{#1}@firstpl\endcsname
7679 }

```

\showgloftype \showgloftype{\label{}}

```

7680 \newcommand*{\showgloftype}[1]{%
7681   \expandafter\show\csname glo\glsdetoklabel{#1}@type\endcsname
7682 }

```

\showglocounter \showglocounter{\label{}}

```

7683 \newcommand*{\showglocounter}[1]{%
7684   \expandafter\show\csname glo\glsdetoklabel{#1}@counter\endcsname
7685 }

```

\showglouserii \showglouserii{\label{}}

```

7686 \newcommand*{\showglouserii}[1]{%
7687   \expandafter\show\csname glo\glsdetoklabel{#1}@userii\endcsname
7688 }

```

\showglouseriii \showglouseriii{\label{}}

```

7689 \newcommand*{\showglouseriii}[1]{%
7690   \expandafter\show\csname glo\glsdetoklabel{#1}@useriii\endcsname
7691 }

```

\showglouseriiii \showglouseriiii{\label{}}


```

7692 \newcommand*{\showglouseriii}[1]{%
7693   \expandafter\show\csname glo\glsdetoklabel{#1}@useriii\endcsname
7694 }

```

\showglouseriv \showglouseriv{<label>}

```

7695 \newcommand*{\showglouseriv}[1]{%
7696   \expandafter\show\csname glo\glsdetoklabel{#1}@useriv\endcsname
7697 }

```

\showglouserv \showglouserv{<label>}

```

7698 \newcommand*{\showglouserv}[1]{%
7699   \expandafter\show\csname glo\glsdetoklabel{#1}@userv\endcsname
7700 }

```

\showglouservi \showglouservi{<label>}

```

7701 \newcommand*{\showglouservi}[1]{%
7702   \expandafter\show\csname glo\glsdetoklabel{#1}@uservi\endcsname
7703 }

```

\showgloname \showgloname{<label>}

```

7704 \newcommand*{\showgloname}[1]{%
7705   \expandafter\show\csname glo\glsdetoklabel{#1}@name\endcsname
7706 }

```

\showglodesc \showglodesc{<label>}

```

7707 \newcommand*{\showglodesc}[1]{%
7708   \expandafter\show\csname glo\glsdetoklabel{#1}@desc\endcsname
7709 }

```

howglodescplural \showglodescplural{<label>}

```

7710 \newcommand*{\showglodescplural}[1]{%
7711   \expandafter\show\csname glo@glstdetoklabel{#1}@descplural\endcsname
7712 }

```

\showglosort \showglosort{<label>}

```

7713 \newcommand*{\showglosort}[1]{%
7714   \expandafter\show\csname glo@glstdetoklabel{#1}@sort\endcsname
7715 }

```

\showglosymbol \showglosymbol{<label>}

```

7716 \newcommand*{\showglosymbol}[1]{%
7717   \expandafter\show\csname glo@glstdetoklabel{#1}@symbol\endcsname
7718 }

```

wglosymbolplural \showglosymbolplural{<label>}

```

7719 \newcommand*{\showglosymbolplural}[1]{%
7720   \expandafter\show\csname glo@glstdetoklabel{#1}@symbolplural\endcsname
7721 }

```

\showgloshort \showgloshort{<label>}

```

7722 \newcommand*{\showgloshort}[1]{%
7723   \expandafter\show\csname glo@glstdetoklabel{#1}@short\endcsname
7724 }

```

\showglolong \showglolong{<label>}

```

7725 \newcommand*{\showglolong}[1]{%
7726   \expandafter\show\csname glo@glstdetoklabel{#1}@long\endcsname
7727 }

```

\showgloindex \showgloindex{<label>}

```

7728 \newcommand*{\showgloindex}[1]{%
7729   \expandafter\show\csname glo@glstetoklabel{#1}@index\endcsname
7730 }

```

\showgloflag \showgloflag{<label>}

```

7731 \newcommand*{\showgloflag}[1]{%
7732   \expandafter\show\csname ifglo@glstetoklabel{#1}@flag\endcsname
7733 }

```

\showgloloclist \showgloloclist{<label>}

```

7734 \newcommand*{\showgloloclist}[1]{%
7735   \expandafter\show\csname glo@glstetoklabel{#1}@loclist\endcsname
7736 }

```

\showglofield \showglofield{<label>}{<field>}

```

7737 \newcommand*{\showglofield}[2]{%
7738   \cshow{glo@glstetoklabel{#1}@#2}%
7739 }

```

showacronymlists \showacronymlists

Show list of glossaries that have been flagged as a list of acronyms.

```

7740 \newcommand*{\showacronymlists}{%
7741   \show\@glsacronymlists
7742 }

```

\showglossaries \showglossaries

Show list of defined glossaries.

```

7743 \newcommand*{\showglossaries}{%
7744   \show\@glo@types
7745 }

```

\showglossaryin \showglossaryin{<glossary-label>}

Show the ‘in’ extension for the given glossary.

```
7746 \newcommand*{\showglossaryin}[1]{%
7747   \expandafter\show\csname @glotype@#1@in\endcsname
7748 }
```

\showglossaryout \showglossaryout{<glossary-label>}

Show the ‘out’ extension for the given glossary.

```
7749 \newcommand*{\showglossaryout}[1]{%
7750   \expandafter\show\csname @glotype@#1@out\endcsname
7751 }
```

showglossarytitle \showglossarytitle{<glossary-label>}

Show the title for the given glossary.

```
7752 \newcommand*{\showglossarytitle}[1]{%
7753   \expandafter\show\csname @glotype@#1@title\endcsname
7754 }
```

wglossarycounter \showglossarycounter{<glossary-label>}

Show the counter for the given glossary.

```
7755 \newcommand*{\showglossarycounter}[1]{%
7756   \expandafter\show\csname @glotype@#1@counter\endcsname
7757 }
```

wglossaryentries \showglossaryentries{<glossary-label>}

Show the list of entry labels for the given glossary.

```
7758 \newcommand*{\showglossaryentries}[1]{%
7759   \expandafter\show\csname glolist@#1\endcsname
7760 }
```

1.21 Compatibility with version 2.07 and below

In order to fix some bugs in v3.0, it was necessary to change the way information is written to the glo file, which also meant a change in the format of the Xindy style file. The compatibility option is meant for documents that use a customised Xindy style file with \noist. With the compatibility option, hopefully xindy will still be able to process the old document, but the bugs will remain. The issues in versions 2.07 and below:

- With xindy, the counter used by the entry was hard-coded into the Xindy style file. This meant that you couldn't use the counter to swap counters.
- With both xindy and makeindex, if used with hyperref and `\theH<counter>` was different to `\thecounter`, the link in the location number would be undefined.

```

7761 \csname ifglscpatible-2.07\endcsname
7762   \RequirePackage{glossaries-compatible-207}
7763 \fi

```

2 Prefix Support (glossaries-prefix Code)

This package provides a means of adding prefixes to your glossary entries. For example, you may want to use “`\gls{<label>}`” on first use but use “`\an\gls{<label>}`” on subsequent use.

```
7764 \NeedsTeXFormat{LaTeX2e}
```

```
7765 \ProvidesPackage{glossaries-prefix}[2017/01/19 v4.29 (NLCT)]
```

Pass all options to glossaries:

```
7766 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
7767 \ProcessOptions
```

Load glossaries:

```
7768 \RequirePackage{glossaries}
```

Add the new keys:

```
7769 \define@key{glossentry}{prefixfirst}{\def\@glo@entryprefixfirst{#1}}%
```

```
7770 \define@key{glossentry}{prefixfirstplural}{\def\@glo@entryprefixfirstplural{#1}}%
```

```
7771 \define@key{glossentry}{prefix}{\def\@glo@entryprefix{#1}}%
```

```
7772 \define@key{glossentry}{prefixplural}{\def\@glo@entryprefixplural{#1}}%
```

Add them to `\@gls@keymap`:

```
7773 \appto\@gls@keymap{,%
```

```
7774   {prefixfirst}{prefixfirst},%
```

```
7775   {prefixfirstplural}{prefixfirstplural},%
```

```
7776   {prefix}{prefix},%
```

```
7777   {prefixplural}{prefixplural}}%
```

```
7778 }
```

Set the default values:

```
7779 \appto\@newglossaryentryprehook{%
```

```
7780   \def\@glo@entryprefix{}}%
```

```
7781   \def\@glo@entryprefixplural{}}%
```

```
7782   \let\@glo@entryprefixfirst\@gls@default@value
```

```
7783   \let\@glo@entryprefixfirstplural\@gls@default@value
```

```
7784 }
```

Set the assignment code:

```
7785 \appto\@newglossaryentryposthook{%
```

```
7786   \gls@assign@field{\@glo@label}{prefix}{\@glo@entryprefix}}%
```

```
7787   \gls@assign@field{\@glo@label}{prefixplural}{\@glo@entryprefixplural}}%
```

If `prefixfirst` has not been supplied, make it the same as `prefix`.

```
7788 \expandafter\gls@assign@field\expandafter
```

```
7789   {\csname glo@\@glo@label @prefix\endcsname}{\@glo@label}{prefixfirst}}%
```

```
7790   {\@glo@entryprefixfirst}}%
```

If prefixfirstplural has not been supplied, make it the same as prefixplural.

```

7791 \expandafter\gls@assign@field\expandafter
7792   {\csname glo@\@glo@label @prefixplural\endcsname}{\@glo@label}%
7793   {prefixfirstplural}{\@glo@entryprefixfirstplural}%
7794 }

```

Define commands to access these fields:

entryprefixfirst

```

7795 \newcommand*{\glsentryprefixfirst}[1]{\csuse{glo@#1@prefixfirst}}

```

entryfirstplural

```

7796 \newcommand*{\glsentryprefixfirstplural}[1]{\csuse{glo@#1@prefixfirstplural}}

```

\glsentryprefix

```

7797 \newcommand*{\glsentryprefix}[1]{\csuse{glo@#1@prefix}}

```

entryprefixplural

```

7798 \newcommand*{\glsentryprefixplural}[1]{\csuse{glo@#1@prefixplural}}

```

Now for the initial upper case variants:

entryprefixfirst

```

7799 \newrobustcmd*{\Glsentryprefixfirst}[1]{%
7800   \protected@edef\@glo@text{\csname glo@#1@prefixfirst\endcsname}%
7801   \xmakefirstuc\@glo@text
7802 }

```

entryfirstplural

```

7803 \newrobustcmd*{\Glsentryprefixfirstplural}[1]{%
7804   \protected@edef\@glo@text{\csname glo@#1@prefixfirstplural\endcsname}%
7805   \xmakefirstuc\@glo@text
7806 }

```

\Glsentryprefix

```

7807 \newrobustcmd*{\Glsentryprefix}[1]{%
7808   \protected@edef\@glo@text{\csname glo@#1@prefix\endcsname}%
7809   \xmakefirstuc\@glo@text
7810 }

```

entryprefixplural

```

7811 \newrobustcmd*{\Glsentryprefixplural}[1]{%
7812   \protected@edef\@glo@text{\csname glo@#1@prefixplural\endcsname}%
7813   \xmakefirstuc\@glo@text
7814 }

```

Define commands to determine if the prefix keys have been set:

\ifglshasprefix

```
7815 \newcommand*{\ifglshasprefix}[3]{%
7816   \ifcempty{glo@#1@prefix}%
7817   {#3}%
7818   {#2}%
7819 }
```

hasprefixplural

```
7820 \newcommand*{\ifglshasprefixplural}[3]{%
7821   \ifcempty{glo@#1@prefixplural}%
7822   {#3}%
7823   {#2}%
7824 }
```

shasprefixfirst

```
7825 \newcommand*{\ifglshasprefixfirst}[3]{%
7826   \ifcempty{glo@#1@prefixfirst}%
7827   {#3}%
7828   {#2}%
7829 }
```

efixfirstplural

```
7830 \newcommand*{\ifglshasprefixfirstplural}[3]{%
7831   \ifcempty{glo@#1@prefixfirstplural}%
7832   {#3}%
7833   {#2}%
7834 }
```

Define commands that insert the prefix before commands like \gls:

\pgls

```
7835 \newrobustcmd{\pgls}{\@gls@hyp@opt\@pgls}
```

\@pgls Unstarred version.

```
7836 \newcommand*{\@pgls}[2][ ]{%
7837   \new@ifnextchar[%
7838   {\@pgls@{#1}{#2}}%
7839   {\@pgls@{#1}{#2}[ ]}%
7840 }
```

\@pgls@ Read in the final optional argument:

```
7841 \def\@pgls@#1#2[#3]{%
7842   \glsdoifexists{#2}%
7843   {%
7844     \ifglsused{#2}%
7845     {%
7846       \glsentryprefix{#2}%
7847     }%

```



```

7848     {%
7849     \glsentryprefixfirst{#2}%
7850     }%
7851     \@gls@{#1}{#2}[#3]%
7852     }%
7853 }

```

Similarly for the plural version:

```

\pglsp1
7854 \newrobustcmd{\pglsp1}{\@gls@hyp@opt\@pglsp1}

```

\@pglsp1 Unstarred version.

```

7855 \newcommand*{\@pglsp1}[2][ ]{%
7856   \new@ifnextchar[%
7857   {\@pglsp1@{#1}{#2}}%
7858   {\@pglsp1@{#1}{#2}[ ]}%
7859 }

```

\@pglsp1@ Read in the final optional argument:

```

7860 \def\@pglsp1@#1#2[#3]{%
7861   \glsdoifexists{#2}%
7862   {%
7863     \ifglsused{#2}%
7864     {%
7865       \glsentryprefixplural{#2}%
7866     }%
7867     {%
7868       \glsentryprefixfirstplural{#2}%
7869     }%
7870     \@glspl@{#1}{#2}[#3]%
7871   }%
7872 }

```

Now for the first letter upper case versions:

```

\Pgls
7873 \newrobustcmd{\Pgls}{\@gls@hyp@opt\@Pgls}

```

\@Pgls Unstarred version.

```

7874 \newcommand*{\@Pgls}[2][ ]{%
7875   \new@ifnextchar[%
7876   {\@Pgls@{#1}{#2}}%
7877   {\@Pgls@{#1}{#2}[ ]}%
7878 }

```

\@Pgls@ Read in the final optional argument:

```

7879 \def\@Pgls@#1#2[#3]{%

```

```

7880 \glsdoifexists{#2}%
7881 {%
7882   \ifglsused{#2}%
7883   {%
7884     \ifglshasprefix{#2}%
7885     {%
7886       \Glsentryprefix{#2}%
7887       \@gls@{#1}{#2}[#3]%
7888     }%
7889     {\@Gls@{#1}{#2}[#3]}%
7890   }%
7891   {%
7892     \ifglshasprefixfirst{#2}%
7893     {%
7894       \Glsentryprefixfirst{#2}%
7895       \@gls@{#1}{#2}[#3]%
7896     }%
7897     {\@Gls@{#1}{#2}[#3]}%
7898   }%
7899 }%
7900 }

```

Similarly for the plural version:

```

\Pglspl
7901 \newrobustcmd{\Pglspl}{\@gls@hyp@opt\@Pglspl}

```

\@Pglspl Unstarred version.

```

7902 \newcommand*{\@Pglspl}[2] [] {%
7903   \new@ifnextchar[%
7904   {\@Pglspl@{#1}{#2}}%
7905   {\@Pglspl@{#1}{#2} []}%
7906 }

```

\@Pglspl@ Read in the final optional argument:

```

7907 \def\@Pglspl@#1#2[#3] {%
7908   \glsdoifexists{#2}%
7909   {%
7910     \ifglsused{#2}%
7911     {%
7912       \ifglshasprefixplural{#2}%
7913       {%
7914         \Glsentryprefixplural{#2}%
7915         \@glspl@{#1}{#2}[#3]%
7916       }%
7917       {\@Glspl@{#1}{#2}[#3]}%
7918     }%
7919     {%
7920       \ifglshasprefixfirstplural{#2}%

```

```

7921      {%
7922      \Glsentryprefixfirstplural{#2}%
7923      \@glsp1@{#1}{#2}[#3]%
7924      }%
7925      {\@Glspl@{#1}{#2}[#3]}%
7926      }%
7927  }%
7928 }

```

Finally the all upper case versions:

\PGLS

```

7929 \newrobustcmd{\PGLS}{\@gls@hyp@opt\PGLS}

```

\@PGLS Unstarred version.

```

7930 \newcommand*{\@PGLS}[2][{}]{%
7931   \new@ifnextchar[%
7932   {\@PGLS@{#1}{#2}}%
7933   {\@PGLS@{#1}{#2}[]}%
7934 }

```

\@PGLS@ Read in the final optional argument:

```

7935 \def\@PGLS@#1#2[#3]{%
7936   \glsdoifexists{#2}%
7937   {%
7938     \ifglsused{#2}%
7939     {%
7940       \mfirstucMakeUppercase{\glsentryprefix{#2}}%
7941     }%
7942     {%
7943       \mfirstucMakeUppercase{\glsentryprefixfirst{#2}}%
7944     }%
7945     \@GLS@{#1}{#2}[#3]%
7946   }%
7947 }

```

Plural version:

\PGLSp1

```

7948 \newrobustcmd{\PGLSp1}{\@gls@hyp@opt\PGLSp1}

```

\@PGLSp1 Unstarred version.

```

7949 \newcommand*{\@PGLSp1}[2][{}]{%
7950   \new@ifnextchar[%
7951   {\@PGLSp1@{#1}{#2}}%
7952   {\@PGLSp1@{#1}{#2}[]}%
7953 }

```

\@PGLSp1@ Read in the final optional argument:

```
7954 \def\@PGLSp1@#1#2[#3]{%  
7955   \glsdoifexists{#2}%  
7956   {%  
7957     \ifglsused{#2}%  
7958     {%  
7959       \mfirstucMakeUppercase{\glsentryprefixplural{#2}}%  
7960     }%  
7961     {%  
7962       \mfirstucMakeUppercase{\glsentryprefixfirstplural{#2}}%  
7963     }%  
7964     \@GLSp1@{#1}{#2}[#3]%  
7965   }%  
7966 }
```

3 Glossary Styles

3.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```
7967 \ProvidesPackage{glossary-hypernav}[2017/01/19 v4.29 (NLCT)]
```

The commands defined in this package are provided to help navigate around the groups within a glossary (see [section 1.16](#).) `\printglossary` (and `\printglossaries`) set `\@glo@type` to the label of the current glossary. This is used to create a unique hypertarget in the event of multiple glossaries.

```
\glsnavhyperlink[⟨type⟩]{⟨label⟩}{⟨text⟩}
```

This command makes `⟨text⟩` a hyperlink to the glossary group whose label is given by `⟨label⟩` for the glossary given by `⟨type⟩`.

`glsnavhyperlink`

```
7968 \newcommand*{\glsnavhyperlink}[3][\@glo@type]{%
7969   \edef\gls@grplabel{#2}\protected@edef\gls@grptitle{#3}%
7970   \@glslink{\glsnavhyperlinkname{#1}{#2}}{#3}}
```

`glsnavhyperlinkname`

Expands to the hypertarget name. The first argument is the glossary type. The second argument is the group label.

```
7971 \newcommand*{\glsnavhyperlinkname}[2]{\glsn:#1@#2}
```

```
\glsnavhypertarget[⟨type⟩]{⟨label⟩}{⟨text⟩}
```

This command makes `⟨text⟩` a hypertarget for the glossary group whose label is given by `⟨label⟩` in the glossary given by `⟨type⟩`. If `⟨type⟩` is omitted, `\@glo@type` is used which is set by `\printglossary` to the current glossary label.

`glsnavhypertarget`

```
7972 \newcommand*{\glsnavhypertarget}[3][\@glo@type]{%
  Add this group to the aux file for re-run check.
7973   \protected@write\auxout{}{\string\@gls@hypergroup{#1}{#2}}%
  Add the target.
7974   \@glstarget{\glsnavhyperlinkname{#1}{#2}}{#3}%
```

Check list of known groups to determine if a re-run is required.

```
7975 \expandafter\let
7976 \expandafter\@gls@list\csname @gls@hypergroup@list@#1\endcsname
```

Iterate through list and terminate loop if this group is found.

```
7977 \@for\@gls@elem:=\@gls@list\do{%
7978 \ifthenelse{\equal{\@gls@elem}{#2}}{\@endfortrue}{}}%
```

Check if list terminated prematurely.

```
7979 \if@endfor
7980 \else
```

This group was not included in the list, so issue a warning.

```
7981 \GlossariesWarningNoLine{Navigation panel
7982 for glossary type ‘#1’^^Jmissing group ‘#2’}%
7983 \gdef\gls@hypergroup@rerun{%
7984 \GlossariesWarningNoLine{Navigation panel
7985 has changed. Rerun LaTeX}}%
7986 \fi
7987 }
```

`hypergroup@rerun` Give a warning at the end if re-run required

```
7988 \let\gls@hypergroup@rerun\relax
7989 \AtEndDocument{\gls@hypergroup@rerun}
```

`@gls@hypergroup` This adds to (or creates) the command `\@gls@hypergroup@list@<glossary type>` which lists all groups for a given glossary, so that the navigation bar only contains those groups that are present. However it requires at least 2 runs to ensure the information is up-to-date.

```
7990 \newcommand*{\@gls@hypergroup}[2]{%
7991 \@ifundefined{\@gls@hypergroup@list@#1}{%
7992 \expandafter\xdef\csname @gls@hypergroup@list@#1\endcsname{#2}%
7993 }{%
7994 \expandafter\let\expandafter\@gls@tmp
7995 \csname @gls@hypergroup@list@#1\endcsname
7996 \expandafter\xdef\csname @gls@hypergroup@list@#1\endcsname{%
7997 \@gls@tmp,#2}%
7998 }%
7999 }
```

The `\glsnavigation` command displays a simple glossary group navigation. The symbol and number elements are defined separately, so that they can be suppressed if need be. Note that this command will produce a link to all 28 groups, but some groups may not be defined if there are groups that do not contain any terms, in which case you will get an undefined hyperlink warning. Now for the whole navigation bit:

`\glsnavigation`

```
8000 \newcommand*{\glsnavigation}{%
8001 \def\@gls@between{}%
8002 \ifcsundef{\@gls@hypergroup@list@\@glo@type}%
```

```

8003 {%
8004   \def\@gls@list{}%
8005 }%
8006 {%
8007   \expandafter\let\expandafter\@gls@list
8008     \csname @gls@hypergroup@list@\@glo@type\endcsname
8009 }%
8010 \@for\@gls@tmp:=\@gls@list\do{%
8011   \@gls@between

8012   \@gls@getgrouptitle{\@gls@tmp}{\@gls@grptitle}%
8013   \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
8014   \let\@gls@between\glshypernavsep
8015 }%
8016 }

```

`\glshypernavsep` Separator for the hyper navigation bar.

```
8017 \newcommand*{\glshypernavsep}{\space\textbar\space}
```

The `\glssymbolnav` produces a simple navigation set of links for just the symbol and number groups. This used to be used at the start of `\glsnavigation`. This command is no longer needed.

`\glssymbolnav`

```

8018 \newcommand*{\glssymbolnav}{%
8019   \glsnavhyperlink{glsymbols}{\glsgetgrouptitle{glsymbols}}%
8020   \glshypernavsep
8021   \glsnavhyperlink{glsnumbers}{\glsgetgrouptitle{glsnumbers}}%
8022   \glshypernavsep
8023 }

```

3.2 In-line Style (glossary-inline.sty)

This defines an in-line style where the entries are comma-separated with just the name and description displayed.

```
8024 \ProvidesPackage{glossary-inline}[2017/01/19 v4.29 (NLCT)]
```

`inline` Define the inline style.

```

8025 \newglossarystyle{inline}{%
    Start of glossary sets up first empty separator between entries. (This is then changed by
    \glossentry)
8026   \renewenvironment{theglossary}%
8027     {%
8028       \def\gls@inlinesep{}%
8029       \def\gls@inlinesubsep{}%
8030       \def\gls@inlinepostchild{}%
8031     }%
8032     {\glspostinline}%

```

No header:

```
8033 \renewcommand*{\glossaryheader}{}%
```

No group headings (if heading is required, add `\glsinlinedopostchild` to start definition in case heading follows a child entry):

```
8034 \renewcommand*{\glsgroupheading}[1]{}%
```

Just display separator followed by name and description:

```
8035 \renewcommand{\glossentry}[2]{%
8036   \glsinlinedopostchild
8037   \gls@inlinesep
8038   \glsentryitem{##1}%
8039   \glsinlinenameformat{##1}{%
8040     \glossentryname{##1}%
8041   }%
8042   \ifglsdescsuppressed{##1}%
8043   {%
8044     \glsinlineemptydescformat
8045     {%
8046       \glossentrysymbol{##1}%
8047     }%
8048     {%
8049       ##2%
8050     }%
8051   }%
8052   {%
8053     \ifglshasdesc{##1}%
8054     {\glsinlinedescformat{\glossentrydesc{##1}}{\glossentrysymbol{##1}}{##2}}%
8055     {\glsinlineemptydescformat{\glossentrysymbol{##1}}{##2}}%
8056   }%
8057   \ifglshaschildren{##1}%
8058   {%
8059     \glsresetsubentrycounter
8060     \glsinlineparentchildseparator
8061     \def\gls@inlinesubsep{}%
8062     \def\gls@inlinepostchild{\glsinlinepostchild}%
8063   }%
8064   {}%
8065   \def\gls@inlinesep{\glsinlineseparator}%
8066 }%
```

Sub-entries display description:

```
8067 \renewcommand{\subglossentry}[3]{%
8068   \gls@inlinesubsep%
8069   \glsinlinesubnameformat{##2}{%
8070     \glossentryname{##2}%
8071   }%
8072   \glsinlinesubdescformat{\glossentrydesc{##2}}{\glossentrysymbol{##2}}{##3}%
8073   \def\gls@inlinesubsep{\glsinlinesubseparator}%
8074 }%
```


Nothing special between groups:

```
8075 \renewcommand*{\glsgroupskip}{}%  
8076 }
```

linedopostchild

```
8077 \newcommand*{\glsinlinedopostchild}{%  
8078     \gls@inlinepostchild  
8079     \def\gls@inlinepostchild{}%  
8080 }
```

inlineseparator Separator to use between entries.

```
8081 \newcommand*{\glsinlineseparator}{;\space}
```

inlinesubseparator Separator to use between sub-entries.

```
8082 \newcommand*{\glsinlinesubseparator}{,\space}
```

parentchildseparator Separator to use between parent and children.

```
8083 \newcommand*{\glsinlineparentchildseparator}{:\space}
```

inlinepostchild Hook to use between child and next entry

```
8084 \newcommand*{\glsinlinepostchild}{}
```

\glspostinline Terminator for inline glossary.

```
8085 \newcommand*{\glspostinline}{\glspostdescription\space}
```

inlinenameformat Formats the name of the entry (first argument label, second argument name):

```
8086 \newcommand*{\glsinlinenameformat}[2]{\glstarget{#1}{#2}}
```

inlinedescformat Formats the entry's description, symbol and location list:

```
8087 \newcommand*{\glsinlinedescformat}[3]{\space#1}
```

emptydescformat Formats the entry's symbol and location list when the description is empty:

```
8088 \newcommand*{\glsinlineemptydescformat}[2]{}
```

inlinesubnameformat Formats the name of the subentry (first argument label, second argument name):

```
8089 \newcommand*{\glsinlinesubnameformat}[2]{\glstarget{#1}{}}
```

inlinesubdescformat Formats the subentry's description, symbol and location list:

```
8090 \newcommand*{\glsinlinesubdescformat}[3]{#1}
```

3.3 List Style (glossary-list.sty)

The style file defines glossary styles that use the description environment. Note that since the entry name is placed in the optional argument to the `\item` command, it will appear in a bold font by default.

```
8091 \ProvidesPackage{glossary-list}[2017/01/19 v4.29 (NLCT)]
```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```

8092 \providecommand{\indexspace}{%
8093   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
8094 }

```

`tgrouphdheaderfmt` Provide a way of adjusting the format of the group headings.

```

8095 \newcommand*{\glslistgrouphdheaderfmt}[1]{#1}

```

`tnavigationitem` Provide a way of adjusting the format of the navigation header. This puts the navigation line inside the optional argument of `item` to prevent unwanted space occurring at the start, but this can cause a problem if the navigation line is too long. With this command, it makes it easier for the user to customise the style without having to remember to modify `\glossaryheader` after the style has been set.

```

8096 \newcommand*{\glslistnavigationitem}[1]{\item[#1]}

```

`list` The list glossary style uses the `description` environment. The group separator `\glsgroupskip` is redefined as `\indexspace` which produces a gap between groups. The glossary heading and the group headings do nothing. Sub-entries immediately follow the main entry without the sub-entry name. This style does not use the entry's symbol. This is used as the default style for the glossaries package.

```

8097 \newglossarystyle{list}{%

```

Use `description` environment:

```

8098   \renewenvironment{theglossary}%
8099     {\begin{description}}{\end{description}}%

```

No header at the start of the environment:

```

8100   \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

8101   \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries start a new item in the list:

```

8102   \renewcommand*{\glossentry}[2]{%
8103     \item[\glsentryitem{##1}%
8104       \glstarget{##1}{\glossentryname{##1}}]
8105     \glossentrydesc{##1}\glspostdescription\space ##2}%

```

Sub-entries continue on the same line:

```

8106   \renewcommand*{\subglossentry}[3]{%
8107     \glssubentryitem{##2}%
8108     \glstarget{##2}{\strut}\space
8109     \glossentrydesc{##2}\glspostdescription\space ##3.}%

```

Add vertical space between groups:

```

8110   \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
8111 }

```

listgroup The listgroup style is like the list style, but the glossary groups have headings.

```
8112 \newglossarystyle{listgroup}{%  
    Base it on the list style:  
8113   \setglossarystyle{list}%  
    Each group has a heading:  
8114   \renewcommand*{\glsgroupheading}[1]{%  
8115     \item[\glslistgroupheaderfmt{\glsgrouptitle{##1}}]}
```

listhypergroup The listhypergroup style is like the listgroup style, but has a set of links to the groups at the start of the glossary.

```
8116 \newglossarystyle{listhypergroup}{%  
    Base it on the list style:  
8117   \setglossarystyle{list}%  
    Add navigation links at the start of the environment.  
8118   \renewcommand*{\glossaryheader}{%  
8119     \glslistnavigationitem{\glsnavigation}}%  
    Each group has a heading with a hypertarget:  
8120   \renewcommand*{\glsgroupheading}[1]{%  
8121     \item[\glslistgroupheaderfmt  
8122       {\glsnavigationhypertarget{##1}{\glsgrouptitle{##1}}}]}
```

altlist The altlist glossary style is like the list style, but places the description on a new line. Sub-entries follow in separate paragraphs without the sub-entry name. This style does not use the entry's symbol.

```
8123 \newglossarystyle{altlist}{%  
    Base it on the list style:  
8124   \setglossarystyle{list}%  
    Main (level 0) entries start a new item in the list with a line break after the entry name:  
8125   \renewcommand*{\glossentry}[2]{%  
8126     \item[\glssentryitem{##1}%  
8127       \glstarget{##1}{\glossentryname{##1}}}%  
    Version 3.04 changed \newline to the following paragraph break stuff (thanks to Daniel Geb-  
    hardt for supplying the fix) to prevent a page break occurring at this point.  
8128     \mbox{}\par\nobreak\@afterheading  
8129     \glossentrydesc{##1}\glspostdescription\space ##2}%  
    Sub-entries start a new paragraph:  
8130   \renewcommand{\subglossentry}[3]{%  
8131     \par  
8132     \glssubentryitem{##2}%  
8133     \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space ##3}%  
8134 }
```

altlistgroup The altlistgroup glossary style is like the altlist style, but the glossary groups have headings.

```
8135 \newglossarystyle{altlistgroup}{%  
    Base it on the altlist style:  
8136 \setglossarystyle{altlist}%  
    Each group has a heading:  
8137 \renewcommand*{\glsgroupheading}[1]{%  
8138 \item[\glslistgroupheaderfmt{\glsgrouptitle{##1}}]}
```

altlisthypergroup The altlisthypergroup glossary style is like the altlistgroup style, but has a set of links to the groups at the start of the glossary.

```
8139 \newglossarystyle{altlisthypergroup}{%  
    Base it on the altlist style:  
8140 \setglossarystyle{altlist}%  
    Add navigation links at the start of the environment.  
8141 \renewcommand*{\glossaryheader}{%  
8142 \glslistnavigationitem{\glsnavigation}}%  
    Each group has a heading with a hypertarget:  
8143 \renewcommand*{\glsgroupheading}[1]{%  
8144 \item[\glslistgroupheaderfmt  
8145 {\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}]}
```

listdotted The listdotted glossary style was supplied by Axel Menzel. I've modified it slightly so that the distance from the start of the name to the end of the dotted line is specified by `\glslistdottedwidth`. Note that this style ignores the page numbers as well as the symbol. Sub-entries are displayed in the same way as top-level entries.

```
8146 \newglossarystyle{listdotted}{%  
    Base it on the list style:  
8147 \setglossarystyle{list}%  
    Each main (level 0) entry starts a new item:  
8148 \renewcommand*{\glossentry}[2]{%  
8149 \item[]\makebox[\glslistdottedwidth][l]{%  
8150 \glssentryitem{##1}%  
8151 \glstarget{##1}{\glossentryname{##1}}%  
8152 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##1}}%  
    Sub entries have the same format as main entries:  
8153 \renewcommand*{\subglossentry}[3]{%  
8154 \item[]\makebox[\glslistdottedwidth][l]{%  
8155 \glssubentryitem{##2}%  
8156 \glstarget{##2}{\glossentryname{##2}}%  
8157 \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}\glossentrydesc{##2}}%  
8158 }
```

listdottedwidth

```
8159 \newlength\glslistdottedwidth
8160 \setlength{\glslistdottedwidth}{.5\hsize}
```

sublistdotted This style is similar to the `glostylelistdotted` style, except that the main entries just have the name displayed.

```
8161 \newglossarystyle{sublistdotted}{%
    Base it on the listdotted style:
8162 \setglossarystyle{listdotted}%
    Main (level 0) entries just display the name:
8163 \renewcommand*{\glossentry}[2]{%
8164 \item[\glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}}]}%
8165 }
```

3.4 Glossary Styles using `longtable` (the `glossary-long` package)

The glossary styles defined in the package used the `longtable` environment in the glossary.

```
8166 \ProvidesPackage{glossary-long}[2017/01/19 v4.29 (NLCT)]
```

Requires the package:

```
8167 \RequirePackage{longtable}
```

`\glsdescwidth` This is a length that governs the width of the description column. (There's a chance that the user may specify `nolong` and then load later, in which case `\glsdescwidth` may have already been defined by . The same goes for `\glspagelistwidth`.)

```
8168 \@ifundefined{glsdescwidth}{%
8169 \newlength\glsdescwidth
8170 \setlength{\glsdescwidth}{0.6\hsize}
8171 }{}
```

`\glspagelistwidth` This is a length that governs the width of the page list column.

```
8172 \@ifundefined{glspagelistwidth}{%
8173 \newlength\glspagelistwidth
8174 \setlength{\glspagelistwidth}{0.1\hsize}
8175 }{}
```

`long` The long glossary style command which uses the `longtable` environment:

```
8176 \newglossarystyle{long}{%
    Use longtable with two columns:
8177 \renewenvironment{theglossary}%
8178 {\begin{longtable}\lp{\glsdescwidth}}%
8179 {\end{longtable}}%
    Do nothing at the start of the environment:
8180 \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
8181 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
8182 \renewcommand{\glossentry}[2]{%
8183   \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8184   \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
8185 }%
```

Sub entries displayed on the following row without the name:

```
8186 \renewcommand{\subglossentry}[3]{%
8187   &
8188   \glssubentryitem{##2}%
8189   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8190   ##3\tabularnewline
8191 }%
```

Blank row between groups: The check for nogroupskip must occur outside `\glsgroupskip`
(<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
8192 \ifglsgroupskip
8193 \renewcommand*{\glsgroupskip}{}%
8194 \else
8195 \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
8196 \fi
8197 }
```

longborder The longborder style is like the above, but with horizontal and vertical lines:

```
8198 \newglossarystyle{longborder}{%
```

Base it on the `glostylelong` style:

```
8199 \setglossarystyle{long}%
```

Use `longtable` with two columns with vertical lines between each column:

```
8200 \renewenvironment{theglossary}{%
8201   \begin{longtable}{|l|p{\glstdescwidth}|}{\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8202 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8203 }
```

longheader The longheader style is like the long style but with a header:

```
8204 \newglossarystyle{longheader}{%
```

Base it on the `glostylelong` style:

```
8205 \setglossarystyle{long}%
```

Set the table's header:

```
8206 \renewcommand*{\glossaryheader}{%
8207   \bfseries \entryname & \bfseries \descriptionname\tabularnewline\endhead}%
8208 }
```

ongheaderborder The longheaderborder style is like the long style but with a header and border:

```
8209 \newglossarystyle{longheaderborder}{%
```

Base it on the glostylelongborder style:

```
8210 \setglossarystyle{longborder}{%
```

Set the table's header and add horizontal line to table's foot:

```
8211 \renewcommand*{\glossaryheader}{%
8212 \hline\bfseries \entryname & \bfseries
8213 \descriptionname\tabularnewline\hline
8214 \endhead
8215 \hline\endfoot}%
8216 }
```

long3col The long3col style is like long but with 3 columns

```
8217 \newglossarystyle{long3col}{%
```

Use a longtable with 3 columns:

```
8218 \renewenvironment{theglossary}%
8219 {\begin{longtable}\lp{\glstdescwidth}p{\glspagelistwidth}}}%
8220 {\end{longtable}}%
```

No table header:

```
8221 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
8222 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8223 \renewcommand{\glossentry}[2]{%
8224 \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8225 \glossentrydesc{##1} & ##2\tabularnewline
8226 }%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
8227 \renewcommand{\subglossentry}[3]{%
8228 &
8229 \glssubentryitem{##2}%
8230 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8231 ##3\tabularnewline
8232 }%
```

Blank row between groups: The check for nogroupskip must occur outside \glsgroupskip (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
8233 \ifglsnogroupskip
8234 \renewcommand*{\glsgroupskip}{}%
8235 \else
8236 \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8237 \fi
8238 }
```

long3colborder The long3colborder style is like the long3col style but with a border:

```
8239 \newglossarystyle{long3colborder}{%
      Base it on the glostylelong3col style:
8240   \setglossarystyle{long3col}%
      Use a longtable with 3 columns with vertical lines around them:
8241   \renewenvironment{theglossary}%
8242     {\begin{longtable}{|l|p{\glstdescwidth}|p{\glspagelistwidth}|}%
8243     {\end{longtable}}}%
      Place horizontal lines at the head and foot of the table:
8244   \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8245 }
```

long3colheader The long3colheader style is like long3col but with a header row:

```
8246 \newglossarystyle{long3colheader}{%
      Base it on the glostylelong3col style:
8247   \setglossarystyle{long3col}%
      Set the table's header:
8248   \renewcommand*{\glossaryheader}{%
8249     \bfseries\entryname&\bfseries\descriptionname&
8250     \bfseries\pagelistname\tabularnewline\endhead}%
8251 }
```

colheaderborder The long3colheaderborder style is like the above but with a border

```
8252 \newglossarystyle{long3colheaderborder}{%
      Base it on the glostylelong3colborder style:
8253   \setglossarystyle{long3colborder}%
      Set the table's header and add horizontal line at table's foot:
8254   \renewcommand*{\glossaryheader}{%
8255     \hline
8256     \bfseries\entryname&\bfseries\descriptionname&
8257     \bfseries\pagelistname\tabularnewline\hline\endhead
8258     \hline\endfoot}%
8259 }
```

long4col The long4col style has four columns where the third column contains the value of the associated symbol key.

```
8260 \newglossarystyle{long4col}{%
      Use a longtable with 4 columns:
8261   \renewenvironment{theglossary}%
8262     {\begin{longtable}{l1l1l}%
8263     {\end{longtable}}}%
      No table header:
8264   \renewcommand*{\glossaryheader}{}%

```


No group headings:

```
8265 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a single row (name in first column, description in second column, symbol in third column, page list in last column):

```
8266 \renewcommand{\glossentry}[2]{%
8267   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8268   \glossentrydesc{##1} &
8269   \glossentrysymbol{##1} &
8270   ##2\tabularnewline
8271 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8272 \renewcommand{\subglossentry}[3]{%
8273   &
8274   \glssubentryitem{##2}%
8275   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8276   \glossentrysymbol{##2} & ##3\tabularnewline
8277 }%
```

Blank row between groups: The check for nogroupskip must occur outside \glsgroupskip (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
8278 \ifglsnogroupskip
8279 \renewcommand*{\glsgroupskip}{}%
8280 \else
8281 \renewcommand*{\glsgroupskip}{ & & & \tabularnewline}%
8282 \fi
8283 }
```

long4colheader The long4colheader style is like long4col but with a header row.

```
8284 \newglossarystyle{long4colheader}{%
```

Base it on the glostylelong4col style:

```
8285 \setglossarystyle{long4col}{%
```

Table has a header:

```
8286 \renewcommand*{\glossaryheader}{%
8287   \bfseries\entryname&\bfseries\descriptionname&
8288   \bfseries \symbolname&
8289   \bfseries\pagelistname\tabularnewline\endhead}%
8290 }
```

long4colborder The long4colborder style is like long4col but with a border.

```
8291 \newglossarystyle{long4colborder}{%
```

Base it on the glostylelong4col style:

```
8292 \setglossarystyle{long4col}{%
```

Use a longtable with 4 columns surrounded by vertical lines:

```
8293 \renewenvironment{theglossary}{%
```

```

8294    {\begin{longtable}{|l|l|l|l|}}%
8295    {\end{longtable}}%

```

Add horizontal lines to the head and foot of the table:

```

8296    \renewcommand*\glossaryheader{\hline\endhead\hline\endfoot}%
8297 }

```

colheaderborder The long4colheaderborder style is like the above but with a border.

```

8298 \newglossarystyle{long4colheaderborder}{%

```

Base it on the glostylelong4col style:

```

8299    \setglossarystyle{long4col}%

```

Use a longtable with 4 columns surrounded by vertical lines:

```

8300    \renewenvironment{theglossary}%
8301    {\begin{longtable}{|l|l|l|l|}}%
8302    {\end{longtable}}%

```

Add table header and horizontal line at the table's foot:

```

8303    \renewcommand*\glossaryheader{%
8304        \hline\bfseries\entryname&\bfseries\descriptionname&
8305        \bfseries \symbolname&
8306        \bfseries\pagelistname\tabularnewline\hline\endhead
8307    \hline\endfoot}%
8308 }

```

altlong4col The altlong4col style is like the long4col style but can have multiline descriptions and page lists.

```

8309 \newglossarystyle{altlong4col}{%

```

Base it on the glostylelong4col style:

```

8310    \setglossarystyle{long4col}%

```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```

8311    \renewenvironment{theglossary}%
8312    {\begin{longtable}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%
8313    {\end{longtable}}%
8314 }

```

altlong4colheader The altlong4colheader style is like altlong4col but with a header row.

```

8315 \newglossarystyle{altlong4colheader}{%

```

Base it on the glostylelong4colheader style:

```

8316    \setglossarystyle{long4colheader}%

```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```

8317    \renewenvironment{theglossary}%
8318    {\begin{longtable}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%
8319    {\end{longtable}}%
8320 }

```

`altlong4colborder` The `altlong4colborder` style is like `altlong4col` but with a border.

```
8321 \newglossarystyle{altlong4colborder}{%
```

Base it on the `glostylelong4colborder` style:

```
8322 \setglossarystyle{long4colborder}{%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8323 \renewenvironment{theglossary}{%
```

```
8324 {\begin{longtable}{|l|p{\glstdescwidth}|l|p{\glspagelistwidth}|}}%
```

```
8325 {\end{longtable}}%
```

```
8326 }
```

`colheaderborder` The `altlong4colheaderborder` style is like the above but with a header as well as a border.

```
8327 \newglossarystyle{altlong4colheaderborder}{%
```

Base it on the `glostylelong4colheaderborder` style:

```
8328 \setglossarystyle{long4colheaderborder}{%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8329 \renewenvironment{theglossary}{%
```

```
8330 {\begin{longtable}{|l|p{\glstdescwidth}|l|p{\glspagelistwidth}|}}%
```

```
8331 {\end{longtable}}%
```

```
8332 }
```

3.5 Glossary Styles using `longtable` and `booktabs` (the `glossary-longbooktabs`) package

The styles here are based on David Carlisle's patch at <http://tex.stackexchange.com/a/56890>

```
8333 \ProvidesPackage{glossary-longbooktabs}[2017/01/19 v4.29 (NLCT)]
```

Requires `booktabs` package:

```
8334 \RequirePackage{booktabs}
```

and the base packages for long styles:

```
8335 \RequirePackage{glossary-long}
```

```
8336 \RequirePackage{glossary-longragged}
```

(`longtable` and `array` loaded by those packages).

`long-booktabs` The `long-booktabs` style is similar to the `longheader` style but uses the `booktabs` rules and patches `longtable` to check for group skip occurring at a page break.

```
8337 \newglossarystyle{long-booktabs}{%
```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```
8338 \glspatchLToutput
```

As with the longheader style, use the long style as a base.

```
8339 \setglossarystyle{long}%
```

Add a header with rules.

```
8340 \renewcommand*{\glossaryheader}{%
8341   \toprule \bfseries \entryname & \bfseries
8342   \descriptionname\tabularnewline\midrule\endhead
8343   \bottomrule\endfoot}%
```

Check for the nogroupskip package option. If there should be a gap between groups, insert the penalty and the vertical space. The check for nogroupskip should occur outside \glsgroupskip to be on the safe side.

```
8344 \ifglsgroupskip
8345   \renewcommand*{\glsgroupskip}{}%
8346 \else
8347   \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8348 \fi
8349 }
```

ng3col-booktabs The long3col-booktabs style is similar to the long3colheader style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8350 \newglossarystyle{long3col-booktabs}{%
```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```
8351 \glspatchLToutput
```

Use the long3col style as a base.

```
8352 \setglossarystyle{long3col}%
```

Add a header with rules.

```
8353 \renewcommand*{\glossaryheader}{%
8354   \toprule \bfseries \entryname &
8355   \bfseries \descriptionname &
8356   \bfseries \pagelistname
8357   \tabularnewline\midrule\endhead
8358   \bottomrule\endfoot}%
```

Check for the nogroupskip package option. If there should be a gap between groups, insert the penalty and the vertical space. The check for nogroupskip should occur outside \glsgroupskip to be on the safe side.

```
8359 \ifglsgroupskip
8360   \renewcommand*{\glsgroupskip}{}%
8361 \else
8362   \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8363 \fi
8364 }
```

ng4col-booktabs The long4col-booktabs style is similar to the long4colheader style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8365 \newglossarystyle{long4col-booktabs}{%
```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```
8366 \glspatchLToutput
```

Use the long4col style as a base.

```
8367 \setglossarystyle{long4col}%
```

Add a header with rules.

```
8368 \renewcommand*{\glossaryheader}{%
8369   \toprule \bfseries \entryname &
8370   \bfseries \descriptionname &
8371   \bfseries \symbolname &
8372   \bfseries \pagelistname
8373   \tabularnewline\midrule\endhead
8374   \bottomrule\endfoot}%
```

Check for the nogroupskip package option. If there should be a gap between groups, insert the penalty and the vertical space. The check for nogroupskip should occur outside \glsgroupskip to be on the safe side.

```
8375 \ifglsgroupskip
8376   \renewcommand*{\glsgroupskip}{}%
8377 \else
8378   \renewcommand*{\glsgroupskip}{\glspenaltygroupskip}%
8379 \fi
8380 }
```

long4col-booktabs The altlong4col-booktabs style is similar to the altlong4colheader style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8381 \newglossarystyle{altlong4col-booktabs}{%
```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```
8382 \glspatchLToutput
```

Use the long4col-booktabs style as a base.

```
8383 \setglossarystyle{long4col-booktabs}%
```

Change the column specifications:

```
8384 \renewenvironment{theglossary}%
8385   {\begin{longtable}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%
8386   {\end{longtable}}%
8387 }
```

Ragged styles.

longragged-booktabs The longragged-booktabs style is similar to the longragged style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8388 \newglossarystyle{longragged-booktabs}{%
```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```
8389 \glspatchLToutput
```

Use the long-booktabs style as a base.

```
8390 \setglossarystyle{long-booktabs}%
```

Adjust the column specification.

```
8391 \renewenvironment{theglossary}%  
8392   {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}}}%  
8393   {\end{longtable}}%  
8394 }
```

ed3col-booktabs The longragged3col-booktabs style is similar to the longragged3col style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8395 \newglossarystyle{longragged3col-booktabs}{%
```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```
8396 \glspatchLToutput
```

Use the long3col-booktabs style as a base.

```
8397 \setglossarystyle{long3col-booktabs}%
```

Adjust the column specification.

```
8398 \renewenvironment{theglossary}%  
8399   {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}%  
8400     >{\raggedright}p{\glspagelistwidth}}}%  
8401   {\end{longtable}}%  
8402 }
```

ed4col-booktabs The altlongragged4col-booktabs style is similar to the altlongragged4col style but uses the booktabs rules and patches longtable to check for group skip occurring at a page break.

```
8403 \newglossarystyle{altlongragged4col-booktabs}{%
```

If the style change is scoped, the patch will only have a local effect, which may be useful if it conflicts with other tables in the document.

```
8404 \glspatchLToutput
```

Use the altlong4col-booktabs style as a base.

```
8405 \setglossarystyle{altlong4col-booktabs}%
```

Adjust the column specification.

```
8406 \renewenvironment{theglossary}%  
8407   {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%  
8408     >{\raggedright}p{\glspagelistwidth}}}%  
8409   {\end{longtable}}%  
8410 }
```

sLTpenaltycheck

```
8411 \newcommand*{\glslTpenaltycheck}{%  
8412 \ifnum\outputpenalty=-50\vskip-\normalbaselineskip\relax\fi  
8413 }
```

enaltygroupskip

```
8414 \newcommand{\glspenaltygroupskip}{%  
8415   \noalign{\penalty-50\vskip\normalbaselineskip}}
```

restoreLToutput Provide a way of restoring \LT@output for the user.

```
8416 \let\@gls@org@LT@output\LT@output  
8417 \newcommand*{\glsrestoreLToutput}{\let\LT@output\@gls@org@LT@output}
```

This is David's patch, but I've replaced the hard-coded values with \glsLTpenaltycheck to make it easier to adjust.

lspatchLToutput

```
8418 \newcommand*{\glspatchLToutput}{%  
8419   \renewcommand*{\LT@output}{%  
8420     \ifnum\outputpenalty <-\@Mi  
8421       \ifnum\outputpenalty > -\LT@end@pen  
8422         \LT@err{floats and marginpars not allowed in a longtable}\@ehc  
8423       \else  
8424         \setbox\z@\vbox{\unvbox\@cclv}%  
8425         \ifdim \ht\LT@lastfoot>\ht\LT@foot  
8426           \dimen@\pagegoal  
8427           \advance\dimen@-\ht\LT@lastfoot  
8428           \ifdim\dimen@<\ht\z@  
8429             \setbox\@cclv\vbox{\unvbox\z@\copy\LT@foot\vss}%  
8430             \@makecol  
8431             \@outputpage  
8432             \setbox\z@\vbox{\box\LT@head\glsLTpenaltycheck}%  
8433           \fi  
8434         \fi  
8435         \global\@colroom\@colht  
8436         \global\vsizel\@colht  
8437         {\unvbox\z@\box\ifvoid\LT@lastfoot\LT@foot\else\LT@lastfoot\fi}%  
8438       \fi  
8439     \else  
8440       \setbox\@cclv\vbox{\unvbox\@cclv\copy\LT@foot\vss}%  
8441       \@makecol  
8442       \@outputpage  
8443       \global\vsizel\@colroom  
8444       \copy\LT@head  
8445       \glsLTpenaltycheck  
8446       \nobreak  
8447     \fi  
8448   }%  
8449 }
```

3.6 Glossary Styles using longtable (the glossary-longragged package)

The glossary styles defined in the package used the longtable environment in the glossary and use ragged right formatting for the multiline columns.

8450 \ProvidesPackage{glossary-longragged}[2017/01/19 v4.29 (NLCT)]

Requires the package:

8451 \RequirePackage{array}

Requires the package:

8452 \RequirePackage{longtable}

`\glsdescwidth` This is a length that governs the width of the description column. This may have already been defined.

8453 \@ifundefined{glsdescwidth}{%

8454 \newlength{glsdescwidth}

8455 \setlength{glsdescwidth}{0.6\hsize}

8456 }{}

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

8457 \@ifundefined{glspagelistwidth}{%

8458 \newlength{glspagelistwidth}

8459 \setlength{glspagelistwidth}{0.1\hsize}

8460 }{}

`longragged` The longragged glossary style is like the long but uses ragged right formatting for the description column.

8461 \newglossarystyle{longragged}{%

Use longtable with two columns:

8462 \renewenvironment{theglossary}{%

8463 {\begin{longtable}{l>{\raggedright}p{glsdescwidth}}}%

8464 {\end{longtable}}}%

Do nothing at the start of the environment:

8465 \renewcommand*{\glossaryheader}{}%

No heading between groups:

8466 \renewcommand*{\glsgroupheading}[1]{}%

Main (level 0) entries displayed in a row:

8467 \renewcommand{\glossentry}[2]{%

8468 \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &

8469 \glossentrydesc{##1}\glspostdescription\space ##2%

8470 \tabularnewline

8471 }%

Sub entries displayed on the following row without the name:

```
8472 \renewcommand{\subglossentry}[3]{%
8473     &
8474     \glssubentryitem{##2}%
8475     \glstarget{##2}{\strut}\glossentrydesc{##2}%
8476     \glspostdescription\space ##3%
8477     \tabularnewline
8478 }%
```

Blank row between groups: The check for nogroupskip must occur outside `\glsgroupskip`
(<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
8479 \ifglsnogroupskip
8480 \renewcommand*{\glsgroupskip}{}%
8481 \else
8482 \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
8483 \fi
8484 }
```

`longraggedborder` The `longraggedborder` style is like the above, but with horizontal and vertical lines:

```
8485 \newglossarystyle{longraggedborder}{%
```

Base it on the `glostylelongragged` style:

```
8486 \setglossarystyle{longragged}%
```

Use `longtable` with two columns with vertical lines between each column:

```
8487 \renewenvironment{theglossary}{%
8488 \begin{longtable}{|l|>{\raggedright}p{\glstdescwidth}|}%
8489 {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8490 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
8491 }
```

`longraggedheader` The `longraggedheader` style is like the `longragged` style but with a header:

```
8492 \newglossarystyle{longraggedheader}{%
```

Base it on the `glostylelongragged` style:

```
8493 \setglossarystyle{longragged}%
```

Set the table's header:

```
8494 \renewcommand*{\glossaryheader}{%
8495 \bfseries \entryname & \bfseries \descriptionname
8496 \tabularnewline\endhead}%
8497 }
```

`longraggedheaderborder` The `longraggedheaderborder` style is like the `longragged` style but with a header and border:

```
8498 \newglossarystyle{longraggedheaderborder}{%
```

Base it on the `glostylelongraggedborder` style:

```
8499 \setglossarystyle{longraggedborder}%
```

Set the table's header and add horizontal line to table's foot:

```
8500 \renewcommand*{\glossaryheader}{%
8501 \hline\bfseries \entryname & \bfseries \descriptionname
8502 \tabularnewline\hline
8503 \endhead
8504 \hline\endfoot}%
8505 }
```

`longragged3col` The `longragged3col` style is like `longragged` but with 3 columns

```
8506 \newglossarystyle{longragged3col}{%
```

Use a longtable with 3 columns:

```
8507 \renewenvironment{theglossary}%
8508 {\begin{longtable}{l>{\raggedright}p{\glstdescwidth}%
8509 >{\raggedright}p{\glspagelistwidth}}}%
8510 {\end{longtable}}%
```

No table header:

```
8511 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
8512 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
8513 \renewcommand{\glossentry}[2]{%
8514 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8515 \glossentrydesc{##1} & ##2\tabularnewline
8516 }%
```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
8517 \renewcommand{\subglossentry}[3]{%
8518 &
8519 \glssubentryitem{##2}%
8520 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8521 ##3\tabularnewline
8522 }%
```

Blank row between groups: The check for `nogroupskip` must occur outside `\glsgroupskip` (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
8523 \ifglsnogroupskip
8524 \renewcommand*{\glsgroupskip}{}%
8525 \else
8526 \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
8527 \fi
8528 }
```

`ragged3colborder` The `longragged3colborder` style is like the `longragged3col` style but with a border:

```
8529 \newglossarystyle{longragged3colborder}{%
```

Base it on the `glostylelongragged3col` style:

```
8530 \setglossarystyle{longragged3col}%
```

Use a `longtable` with 3 columns with vertical lines around them:

```
8531 \renewenvironment{theglossary}%  
8532 {\begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|%  
8533 >{\raggedright}p{\glspagelistwidth}|}%  
8534 {\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
8535 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%  
8536 }
```

`ragged3colheader` The `longragged3colheader` style is like `longragged3col` but with a header row:

```
8537 \newglossarystyle{longragged3colheader}{%
```

Base it on the `glostylelongragged3col` style:

```
8538 \setglossarystyle{longragged3col}%
```

Set the table's header:

```
8539 \renewcommand*{\glossaryheader}{%  
8540 \bfseries\entryname&\bfseries\descriptionname&  
8541 \bfseries\pagelistname\tabularnewline\endhead}%  
8542 }
```

`colheaderborder` The `longragged3colheaderborder` style is like the above but with a border

```
8543 \newglossarystyle{longragged3colheaderborder}{%
```

Base it on the `glostylelongragged3colborder` style:

```
8544 \setglossarystyle{longragged3colborder}%
```

Set the table's header and add horizontal line at table's foot:

```
8545 \renewcommand*{\glossaryheader}{%  
8546 \hline  
8547 \bfseries\entryname&\bfseries\descriptionname&  
8548 \bfseries\pagelistname\tabularnewline\hline\endhead  
8549 \hline\endfoot}%  
8550 }
```

`altlongragged4col` The `altlongragged4col` style is like the `altlong4col` style defined in the package, except that ragged right formatting is used for the description and page list columns.

```
8551 \newglossarystyle{altlongragged4col}{%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8552 \renewenvironment{theglossary}%  
8553 {\begin{longtable}{l>{\raggedright}p{\glsgdescwidth}l%  
8554 >{\raggedright}p{\glspagelistwidth}}}%  
8555 {\end{longtable}}%
```

No table header:

```
8556 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8557 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a single row (name in first column, description in second column, symbol in third column, page list in last column):

```
8558 \renewcommand{\glossentry}[2]{%
8559   \glsentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8560   \glossentrydesc{##1} & \glossentrysymbol{##1} &
8561   ##2\tabularnewline
8562 }%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
8563 \renewcommand{\subglossentry}[3]{%
8564   &
8565   \glssubentryitem{##2}%
8566   \glstarget{##2}{\strut}\glossentrydesc{##2} &
8567   \glossentrysymbol{##2} & ##3\tabularnewline
8568 }%
```

Blank row between groups: The check for nogroupskip must occur outside `\glsgroupskip` (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
8569 \ifglsnogroupskip
8570 \renewcommand*{\glsgroupskip}{}%
8571 \else
8572 \renewcommand*{\glsgroupskip}{ & & & \tabularnewline}%
8573 \fi
8574 }
```

`ragged4colheader` The `altlongragged4colheader` style is like `altlongragged4col` but with a header row.

```
8575 \newglossarystyle{altlongragged4colheader}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8576 \setglossarystyle{altlongragged4col}%
```

Use a longtable with 4 columns where the second and last columns may have multiple lines in each row:

```
8577 \renewenvironment{theglossary}%
8578   {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
8579     >{\raggedright}p{\glspagelistwidth}}}%
8580   {\end{longtable}}%
```

Table has a header:

```
8581 \renewcommand*{\glossaryheader}{%
8582   \bfseries\entryname&\bfseries\descriptionname&
8583   \bfseries \symbolname&
8584   \bfseries\pagelistname\tabularnewline\endhead}%
8585 }
```

`ragged4colborder` The `altlongragged4colborder` style is like `altlongragged4col` but with a border.

```
8586 \newglossarystyle{altlongragged4colborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8587 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8588 \renewenvironment{theglossary}%  
8589 {\begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|l|}%  
8590 >{\raggedright}p{\glspagelistwidth}|}}%  
8591 {\end{longtable}}%
```

Add horizontal lines to the head and foot of the table:

```
8592 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%  
8593 }
```

`colheaderborder` The `altlongragged4colheaderborder` style is like the above but with a header as well as a border.

```
8594 \newglossarystyle{altlongragged4colheaderborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
8595 \setglossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
8596 \renewenvironment{theglossary}%  
8597 {\begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|l|}%  
8598 >{\raggedright}p{\glspagelistwidth}|}}%  
8599 {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
8600 \renewcommand*{\glossaryheader}{%  
8601 \hline\bfseries\entryname&\bfseries\descriptionname&  
8602 \bfseries \symbolname&  
8603 \bfseries\pagelistname\tabularnewline\hline\endhead  
8604 \hline\endfoot}%  
8605 }
```

3.7 Glossary Styles using multicol (glossary-mcols.sty)

The style file defines glossary styles that use the `multicol` package. These use the tree-like glossary styles in a `multicol` environment.

```
8606 \ProvidesPackage{glossary-mcols}[2017/01/19 v4.29 (NLCT)]
```

Required packages:

```
8607 \RequirePackage{multicol}  
8608 \RequirePackage{glossary-tree}
```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```
8609 \providecommand{\indexspace}{%  
8610 \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax  
8611 }
```

`\glsmcols` Define macro in which to store the number of columns. (Defaults to 2.)

```
8612 \newcommand*{\glsmcols}{2}
```

`mcolindex` Multi-column index style. Same as the index, but puts the glossary in multiple columns. (Ideally the glossary title should go in the optional argument of multicols, but the title isn't part of the glossary style.)

```
8613 \newglossarystyle{mcolindex}{%
8614   \setglossarystyle{index}%
8615   \renewenvironment{theglossary}%
8616     {%
8617       \begin{multicols}{\glsmcols}
8618       \setlength{\parindent}{0pt}%
8619       \setlength{\parskip}{0pt plus 0.3pt}%
8620       \let\item\glstreeitem
8621       \let\subitem\glstreesubitem
8622       \let\subsubitem\glstreesubsubitem
8623     }%
8624     {\end{multicols}}}%
8625 }
```

`mcolindexgroup` As `mcolindex` but has headings:

```
8626 \newglossarystyle{mcolindexgroup}{%
8627   \setglossarystyle{mcolindex}%
8628   \renewcommand*{\glsgroupheading}[1]{%
8629     \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}\indexspace}%
8630 }
```

`indexhypergroup` The `mcolindexhypergroup` style is like the `mcolindexgroup` style but has hyper navigation.

```
8631 \newglossarystyle{mcolindexhypergroup}{%
```

Base it on the `glostylemcolindex` style:

```
8632   \setglossarystyle{mcolindex}%
```

Put navigation links to the groups at the start of the glossary:

```
8633   \renewcommand*{\glossaryheader}{%
8634     \item\glstreenavigationfmt{\glsnavigation}\indexspace}%
```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```
8635   \renewcommand*{\glsgroupheading}[1]{%
8636     \item\glstreegroupheaderfmt
8637       {\glsnavigationtarget{##1}{\glsgrouptitle{##1}}}%
8638     \indexspace}%
8639 }
```

`colindexspannav` Similar to `mcolindexhypergroup`, but puts the navigation line in the optional argument of multicols.

```

8640 \newglossarystyle{mcolindexspannav}{%
8641   \setglossarystyle{index}%
8642   \renewenvironment{theglossary}%
8643     {%
8644       \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glsnavigation}]
8645       \setlength{\parindent}{0pt}%
8646       \setlength{\parskip}{0pt plus 0.3pt}%
8647       \let\item\glstreeitem}%
8648     {\end{multicols}}}%

```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```

8649 \renewcommand*{\glsgroupheading}[1]{%
8650   \item\glstreegroupheaderfmt
8651     {\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}%
8652   \indexspace}%
8653 }

```

mcoltree Multi-column index style. Same as the tree, but puts the glossary in multiple columns.

```

8654 \newglossarystyle{mcoltree}{%
8655   \setglossarystyle{tree}%
8656   \renewenvironment{theglossary}%
8657     {%
8658       \begin{multicols}{\glsmcols}
8659       \setlength{\parindent}{0pt}%
8660       \setlength{\parskip}{0pt plus 0.3pt}%
8661     }%
8662   {\end{multicols}}}%
8663 }

```

mcoltreegroup Like the mcoltree style but the glossary groups have headings.

```

8664 \newglossarystyle{mcoltreegroup}{%
  Base it on the glostylemcoltree style:
8665   \setglossarystyle{mcoltree}%
  Each group has a heading (in bold) followed by a vertical gap):
8666   \renewcommand{\glsgroupheading}[1]{\par
8667     \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par\indexspace}%
8668 }

```

1treehypergroup The mcoltreehypergroup style is like the treegroup style, but has a set of links to the groups at the start of the glossary.

```

8669 \newglossarystyle{mcoltreehypergroup}{%
  Base it on the glostylemcoltree style:
8670   \setglossarystyle{mcoltree}%

```

Put navigation links to the groups at the start of the theglossary environment:

8671 \renewcommand*{\glossaryheader}{\%

8672 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}\%

Each group has a heading (in bold with a target) followed by a vertical gap):

```

8673 \renewcommand*{\glsgroupheading}[1]{%
8674   \par\noindent
8675   \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8676   \indexspace}%
8677 }

```

<code>mcoltreesspannav</code>	Similar to the <code>mcoltreehypergroup</code> style but the navigation line is put in the optional argument of the <code>multicols</code> environment.
-------------------------------	---

```

8678 \newglossarystyle{mcoltreesspannav}{%
8679   \setglossarystyle{tree}%
8680   \renewenvironment{theglossary}%
8681   {%
8682     \begin{multicols}{\glsmcols}[\noindent\glstreenavigationfmt{\glsnavigation}]
8683     \setlength{\parindent}{0pt}%
8684     \setlength{\parskip}{0pt plus 0.3pt}%
8685   }%
8686   {\end{multicols}}%

```

Each group has a heading (in bold with a target) followed by a vertical gap):

```

8687 \renewcommand*{\glsgroupheading}[1]{%
8688   \par\noindent
8689   \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8690   \indexspace}%
8691 }

```

<code>mcoltreenoname</code>	Multi-column index style. Same as the <code>treenoname</code> , but puts the glossary in multiple columns.
-----------------------------	--

```

8692 \newglossarystyle{mcoltreename}{%
8693   \setglossarystyle{treename}%
8694   \renewenvironment{theglossary}%
8695   {%
8696     \begin{multicols}{\glsmcols}
8697     \setlength{\parindent}{0pt}%
8698     \setlength{\parskip}{0pt plus 0.3pt}%
8699   }%
8700   {\end{multicols}}%
8701 }

```

treenonamegroup	Like the mcoltreenoname style but the glossary groups have headings.
-----------------	--

8702 \newglossarystyle{mcoltreenonamegroup}{%

Base it on the `glostylemcoltreenoname` style:

```
8703 \setglossarystyle{mcoltreename}%
```


Give each group a heading:

```
8704 \renewcommand{\glsgroupheading}[1]{\par
8705 \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par\indexspace}%
8706 }
```

`onamehypergroup` The `mcoltreenonamehypergroup` style is like the `mcoltreenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```
8707 \newglossarystyle{mcoltreenonamehypergroup}{%
```

Base it on the `glostylemcoltreenoname` style:

```
8708 \setglossarystyle{mcoltreenoname}%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
8709 \renewcommand*{\glossaryheader}{%
8710 \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8711 \renewcommand*{\glsgroupheading}[1]{%
8712 \par\noindent
8713 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
8714 \indexspace}%
8715 }
```

`eenonamespannav` Similar to the `mcoltreenonamehypergroup` style but the navigation line is put in the optional argument of the `multicols` environment.

```
8716 \newglossarystyle{mcoltreenonamespannav}{%
8717 \setglossarystyle{treenoname}%
8718 \renewenvironment{theglossary}%
8719 {%
8720 \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glsnavigation}]
8721 \setlength{\parindent}{0pt}%
8722 \setlength{\parskip}{0pt plus 0.3pt}%
8723 }%
8724 {\end{multicols}}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
8725 \renewcommand*{\glsgroupheading}[1]{%
8726 \par\noindent
8727 \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
8728 \indexspace}%
8729 }
```

`mcolalttree` Multi-column index style. Same as the `alttree`, but puts the glossary in multiple columns.

```
8730 \newglossarystyle{mcolalttree}{%
8731 \setglossarystyle{alttree}%
8732 \renewenvironment{theglossary}%
8733 {%
8734 \begin{multicols}{\glsmcols}
8735 \def\@gls@prevlevel{-1}%
```

```

8736     \mbox{}\par
8737   }%
8738   {\par\end{multicols}}}%
8739 }

```

colalmtreegroup Like the mcolalmtree style but the glossary groups have headings.

```

8740 \newglossarystyle{mcolalmtreegroup}{%
      Base it on the glostylemcolalmtree style:
8741   \setglossarystyle{mcolalmtree}%
      Give each group a heading.
8742   \renewcommand{\glsgroupheading}[1]{\par
8743     \def\@gls@prevlevel{-1}%
8744     \hangindent0pt\relax
8745     \parindent0pt\relax
8746     \glstreegroupheaderfmt{\glsgrouptitle{##1}}\par\indexspace}%
8747 }

```

treehypergroup The mcolalmtreehypergroup style is like the mcolalmtreegroup style, but has a set of links to the groups at the start of the glossary.

```

8748 \newglossarystyle{mcolalmtreehypergroup}{%
      Base it on the glostylemcolalmtree style:
8749   \setglossarystyle{mcolalmtree}%
      Put the navigation links in the header
8750   \renewcommand*{\glossaryheader}{%
8751     \par
8752     \def\@gls@prevlevel{-1}%
8753     \hangindent0pt\relax
8754     \parindent0pt\relax
8755     \glstreenavigationfmt{\glsnavigation}\par\indexspace}%
      Put a hypertarget at the start of each group
8756   \renewcommand*{\glsgroupheading}[1]{%
8757     \par
8758     \def\@gls@prevlevel{-1}%
8759     \hangindent0pt\relax
8760     \parindent0pt\relax
8761     \glstreegroupheaderfmt{\glssnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
8762     \indexspace}%
8763 }

```

almtreespannav Similar to the mcolalmtreehypergroup style but the navigation line is put in the optional argument of the multicols environment.

```

8764 \newglossarystyle{mcolalmtreespannav}{%
8765   \setglossarystyle{almtree}%
8766   \renewenvironment{theglossary}%
8767   {%
8768     \begin{multicols}{\glsmcols}\noindent\glstreenavigationfmt{\glsnavigation}]

```

```

8769     \def\@gls@prevlevel{-1}%
8770     \mbox{}\par
8771 }%
8772 {\par\end{multicols}}}%

Put a hypertarget at the start of each group
8773 \renewcommand*\@glsgroupheading[1]{%
8774   \par
8775   \def\@gls@prevlevel{-1}%
8776   \hangindent0pt\relax
8777   \parindent0pt\relax
8778   \glstreegroupheaderfmt{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}\par
8779   \indexspace}
8780 }

```

3.8 Glossary Styles using supertabular environment (glossary-super package)

The glossary styles defined in the package use the supertabular environment.

```

8781 \ProvidesPackage{glossary-super}[2017/01/19 v4.29 (NLCT)]

```

Requires the package:

```

8782 \RequirePackage{supertabular}

```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined if has been loaded.

```

8783 \@ifundefined{glsdescwidth}{%
8784   \newlength\glsdescwidth
8785   \setlength{\glsdescwidth}{0.6\hsize}
8786 }{}

```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined if has been loaded.

```

8787 \@ifundefined{glspagelistwidth}{%
8788   \newlength\glspagelistwidth
8789   \setlength{\glspagelistwidth}{0.1\hsize}
8790 }{}

```

`super` The super glossary style uses the supertabular environment (it uses lengths defined in the package.)

```

8791 \newglossarystyle{super}{%

```

Put the glossary in a supertabular environment with two columns and no head or tail:

```

8792   \renewenvironment{theglossary}%
8793     {\tablehead{}\tabletail{}}%
8794     \begin{supertabular}{lp{\glsdescwidth}}}%
8795     {\end{supertabular}}}%

```

Do nothing at the start of the table:

```
8796 \renewcommand*\glossaryheader{}\%
```

No group headings:

```
8797 \renewcommand*\glsgroupheading[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
8798 \renewcommand\glossentry[2]{%
8799   \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8800   \glossentrydesc{##1}\glspostdescription\space ##2\tabularnewline
8801 }%
```

Sub entries put in a row (no name, description and page list in second column):

```
8802 \renewcommand\subglossentry[3]{%
8803   &
8804   \glssubentryitem{##2}%
8805   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
8806   ##3\tabularnewline
8807 }%
```

Blank row between groups: The check for nogroupskip must occur outside `\glsgroupskip` (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
8808 \ifglsgroupskip
8809   \renewcommand*\glsgroupskip{}\%
8810 \else
8811   \renewcommand*\glsgroupskip{& \tabularnewline}%
8812 \fi
8813 }
```

superborder The superborder style is like the above, but with horizontal and vertical lines:

```
8814 \newglossarystyle{superborder}{%
```

Base it on the `glostylesuper` style:

```
8815 \setglossarystyle{super}%
```

Put the glossary in a `supertabular` environment with two columns and a horizontal line in the head and tail:

```
8816 \renewenvironment{theglossary}%
8817   {\tablehead{\hline}\tabletail{\hline}%
8818   \begin{supertabular}{|l|p{\glsglwidth}|}%
8819   {\end{supertabular}}%
8820 }
```

superheader The superheader style is like the super style, but with a header:

```
8821 \newglossarystyle{superheader}{%
```

Base it on the `glostylesuper` style:

```
8822 \setglossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns, a header and no tail:

```

8823 \renewenvironment{theglossary}%
8824   {\tablehead{\bfseries \entryname &
8825     \bfseries\descriptionname\tabularnewline}%
8826     \tabletail{}}%
8827   \begin{supertabular}{lp{\glsgdescwidth}}%
8828     {\end{supertabular}}%
8829 }
```

superheaderborder The superheaderborder style is like the super style but with a header and border:

```
8830 \newglossarystyle{superheaderborder}{%
```

Base it on the glostylessuper style:

```
8831 \setglossarystyle{super}%
```

Put the glossary in a supertabular environment with two columns, a header and horizontal lines above and below the table:

```

8832 \renewenvironment{theglossary}%
8833   {\tablehead{\hline\bfseries \entryname &
8834     \bfseries \descriptionname\tabularnewline\hline}%
8835     \tabletail{\hline}}
8836   \begin{supertabular}{llp{\glsgdescwidth}}%
8837     {\end{supertabular}}%
8838 }
```

super3col The super3col style is like the super style, but with 3 columns:

```
8839 \newglossarystyle{super3col}{%
```

Put the glossary in a supertabular environment with three columns and no head or tail:

```

8840 \renewenvironment{theglossary}%
8841   {\tablehead{}\tabletail{}}%
8842   \begin{supertabular}{lp{\glsgdescwidth}p{\glspagelistwidth}}%
8843     {\end{supertabular}}%
```

Do nothing at the start of the table:

```
8844 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
8845 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```

8846 \renewcommand{\glossentry}[2]{%
8847   \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8848   \glossentrydesc{##1} & ##2\tabularnewline
8849   }%
```

Sub entries on a row (no name, description in second column, page list in last column):

```

8850 \renewcommand{\subglossentry}[3]{%
8851   &
8852   \glssubentryitem{##2}%
```

```

8853      \glstarget{##2}{\strut}\glossentrydesc{##2} &
8854      ##3\tabularnewline
8855    }%

```

Blank row between groups: The check for nogroupskip must occur outside \glsgroupskip (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```

8856  \ifglsnogroupskip
8857    \renewcommand*{\glsgroupskip}{}%
8858  \else
8859    \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
8860  \fi
8861 }

```

super3colborder The super3colborder style is like the super3col style, but with a border:

```
8862 \newglossarystyle{super3colborder}{%
```

Base it on the glostylesuper3col style:

```
8863 \setglossarystyle{super3col}%
```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```

8864 \renewenvironment{theglossary}%
8865   {\tablehead{\hline}\tabletail{\hline}%
8866    \begin{supertabular}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}%
8867    {\end{supertabular}}%
8868 }

```

super3colheader The super3colheader style is like the super3col style but with a header row:

```
8869 \newglossarystyle{super3colheader}{%
```

Base it on the glostylesuper3col style:

```
8870 \setglossarystyle{super3col}%
```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```

8871 \renewenvironment{theglossary}%
8872   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8873    \bfseries\pagelistname\tabularnewline}\tabletail{}}%
8874   \begin{supertabular}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
8875   {\end{supertabular}}%
8876 }

```

colheaderborder The super3colheaderborder style is like the super3col style but with a header and border:

```
8877 \newglossarystyle{super3colheaderborder}{%
```

Base it on the glostylesuper3colborder style:

```
8878 \setglossarystyle{super3colborder}%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```

8879 \renewenvironment{theglossary}%
8880   {\tablehead{\hline

```

```

8881      \bfseries\entryname&\bfseries\descriptionname&
8882      \bfseries\pagelistname\tabularnewline\hline}%
8883      \tabletail{\hline}%
8884      \begin{supertabular}{|l|p{\glsdescwidth}|p{\glspagelistwidth}|}%
8885      {\end{supertabular}}}%
8886 }

```

super4col The super4col glossary style has four columns, where the third column contains the value of the corresponding symbol key used when that entry was defined.

```

8887 \newglossarystyle{super4col}{%

```

Put the glossary in a supertabular environment with four columns and no head or tail:

```

8888 \renewenvironment{theglossary}%
8889 {\tablehead{}\tabletail}%
8890 \begin{supertabular}{|l|l|l|l|}%
8891 \end{supertabular}}%

```

Do nothing at the start of the table:

```

8892 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

8893 \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries on a row with the name in the first column, description in second column, symbol in third column and page list in last column:

```

8894 \renewcommand{\glossentry}[2]{%
8895 \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8896 \glossentrydesc{##1} &
8897 \glossentrysymbol{##1} & ##2\tabularnewline
8898 }%

```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```

8899 \renewcommand{\subglossentry}[3]{%
8900 &
8901 \glssubentryitem{##2}%
8902 \glstarget{##2}{\strut}\glossentrydesc{##2} &
8903 \glossentrysymbol{##2} & ##3\tabularnewline
8904 }%

```

Blank row between groups: The check for nogroupskip must occur outside \glsgroupskip (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```

8905 \ifglsnogroupskip
8906 \renewcommand*{\glsgroupskip}{}%
8907 \else
8908 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
8909 \fi
8910 }

```

super4colheader The super4colheader style is like the super4col but with a header row.

```

8911 \newglossarystyle{super4colheader}{%

```

Base it on the `glostylesuper4col` style:

```
8912 \setglossarystyle{super4col}%
```

Put the glossary in a `supertabular` environment with four columns, a header and no tail:

```
8913 \renewenvironment{theglossary}%
8914 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
8915 \bfseries\symbolname &
8916 \bfseries\pagelistname\tabularnewline}%
8917 \tabletail{}}%
8918 \begin{supertabular}{|l|l|l|l|}%
8919 {\end{supertabular}}%
8920 }
```

`super4colborder` The `super4colborder` style is like the `super4col` but with a border.

```
8921 \newglossarystyle{super4colborder}{%
```

Base it on the `glostylesuper4col` style:

```
8922 \setglossarystyle{super4col}%
```

Put the glossary in a `supertabular` environment with four columns and a horizontal line in the head and tail:

```
8923 \renewenvironment{theglossary}%
8924 {\tablehead{\hline}\tabletail{\hline}%
8925 \begin{supertabular}{|l|l|l|l|}%
8926 {\end{supertabular}}%
8927 }
```

`colheaderborder` The `super4colheaderborder` style is like the `super4col` but with a header and border.

```
8928 \newglossarystyle{super4colheaderborder}{%
```

Base it on the `glostylesuper4col` style:

```
8929 \setglossarystyle{super4col}%
```

Put the glossary in a `supertabular` environment with four columns and a header bordered by horizontal lines and a horizontal line in the tail:

```
8930 \renewenvironment{theglossary}%
8931 {\tablehead{\hline\bfseries\entryname&\bfseries\descriptionname&
8932 \bfseries\symbolname &
8933 \bfseries\pagelistname\tabularnewline\hline}%
8934 \tabletail{\hline}%
8935 \begin{supertabular}{|l|l|l|l|}%
8936 {\end{supertabular}}%
8937 }
```

`altsuper4col` The `altsuper4col` glossary style is like `super4col` but has provision for multiline descriptions.

```
8938 \newglossarystyle{altsuper4col}{%
```

Base it on the `glostylesuper4col` style:

```
8939 \setglossarystyle{super4col}%
```


Put the glossary in a supertabular environment with four columns and no head or tail:

```
8940 \renewenvironment{theglossary}%  
8941 {\tablehead{}\tabletail{}}%  
8942 \begin{supertabular}{lp{\glsdescwidth}lp{\glspagelistwidth}}%  
8943 {\end{supertabular}}%  
8944 }
```

`super4colheader` The `altsuper4colheader` style is like the `altsuper4col` but with a header row.

```
8945 \newglossarystyle{altsuper4colheader}{%
```

Base it on the `glostylesuper4colheader` style:

```
8946 \setglossarystyle{super4colheader}%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```
8947 \renewenvironment{theglossary}%  
8948 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&  
8949 \bfseries\symbolname &  
8950 \bfseries\pagelistname\tabularnewline}\tabletail{}}%  
8951 \begin{supertabular}{lp{\glsdescwidth}lp{\glspagelistwidth}}%  
8952 {\end{supertabular}}%  
8953 }
```

`super4colborder` The `altsuper4colborder` style is like the `altsuper4col` but with a border.

```
8954 \newglossarystyle{altsuper4colborder}{%
```

Base it on the `glostylesuper4colborder` style:

```
8955 \setglossarystyle{super4colborder}%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```
8956 \renewenvironment{theglossary}%  
8957 {\tablehead{\hline}\tabletail{\hline}%  
8958 \begin{supertabular}%  
8959 {\lllp{\glsdescwidth}\lllp{\glspagelistwidth}\ll}%  
8960 {\end{supertabular}}%  
8961 }
```

`colheaderborder` The `altsuper4colheaderborder` style is like the `altsuper4col` but with a header and border.

```
8962 \newglossarystyle{altsuper4colheaderborder}{%
```

Base it on the `glostylesuper4colheaderborder` style:

```
8963 \setglossarystyle{super4colheaderborder}%
```

Put the glossary in a supertabular environment with four columns and a header bordered by horizontal lines and a horizontal line in the tail:

```
8964 \renewenvironment{theglossary}%  
8965 {\tablehead{\hline  
8966 \bfseries\entryname &  
8967 \bfseries\descriptionname &  
8968 \bfseries\symbolname &  
8969 \bfseries\pagelistname\tabularnewline\hline}%
```

```

8970 \tabletail{\hline}%
8971 \begin{supertabular}%
8972   {l|p{\glsdescwidth}|l|p{\glspagelistwidth}|}%
8973 {\end{supertabular}}%
8974 }

```

3.9 Glossary Styles using supertabular environment (glossary-superragged package)

The glossary styles defined in the package use the supertabular environment. These styles are like those provided by the package, except that the multiline columns have ragged right justification.

```

8975 \ProvidesPackage{glossary-superragged}[2017/01/19 v4.29 (NLCT)]

```

Requires the package:

```

8976 \RequirePackage{array}

```

Requires the package:

```

8977 \RequirePackage{supertabular}

```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined.

```

8978 \@ifundefined{glsdescwidth}{%
8979   \newlength{glsdescwidth
8980   \setlength{glsdescwidth}{0.6\hsize}
8981 }{}

```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```

8982 \@ifundefined{glspagelistwidth}{%
8983   \newlength{glspagelistwidth
8984   \setlength{glspagelistwidth}{0.1\hsize}
8985 }{}

```

`superragged` The superragged glossary style uses the supertabular environment.

```

8986 \newglossarystyle{superragged}{%

```

Put the glossary in a supertabular environment with two columns and no head or tail:

```

8987 \renewenvironment{theglossary}%
8988 {\tablehead{}\tabletail{}}%
8989 \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}}%
8990 {\end{supertabular}}%

```

Do nothing at the start of the table:

```

8991 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

8992 \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```

8993 \renewcommand{\glossentry}[2]{%
8994   \glentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
8995   \glossentrydesc{##1}\glspostdescription\space ##2%
8996   \tabularnewline
8997 }%

```

Sub entries put in a row (no name, description and page list in second column):

```

8998 \renewcommand{\subglossentry}[3]{%
8999   &
9000   \glssubentryitem{##2}%
9001   \glstarget{##2}{\strut}\glossentrydesc{##2}\glspostdescription\space
9002   ##3%
9003   \tabularnewline
9004 }%

```

Blank row between groups: The check for nogroupskip must occur outside \glsgroupskip (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```

9005 \ifglsgroupskip
9006   \renewcommand*{\glsgroupskip}{}%
9007 \else
9008   \renewcommand*{\glsgroupskip}{& \tabularnewline}%
9009 \fi
9010 }

```

superraggedborder The superraggedborder style is like the above, but with horizontal and vertical lines:

```

9011 \newglossarystyle{superraggedborder}{%

```

Base it on the glostylessuperragged style:

```

9012 \setglossarystyle{superragged}%

```

Put the glossary in a supertabular environment with two columns and a horizontal line in the head and tail:

```

9013 \renewenvironment{theglossary}%
9014   {\tablehead{\hline}\tabletail{\hline}%
9015   \begin{supertabular}{|l|>{\raggedright}p{\glsglwidth}}}%
9016   {\end{supertabular}}%
9017 }

```

superraggedheader The superraggedheader style is like the super style, but with a header:

```

9018 \newglossarystyle{superraggedheader}{%

```

Base it on the glostylessuperragged style:

```

9019 \setglossarystyle{superragged}%

```

Put the glossary in a supertabular environment with two columns, a header and no tail:

```

9020 \renewenvironment{theglossary}%
9021   {\tablehead{\bfseries \entryname & \bfseries \descriptionname
9022   \tabularnewline}%
9023   \tabletail{}}%

```

```

9024 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}}}%
9025 {\end{supertabular}}}%
9026 }

```

gedheaderborder The superraggedheaderborder style is like the superragged style but with a header and border:

```

9027 \newglossarystyle{superraggedheaderborder}{%

```

Base it on the glostylesuper style:

```

9028 \setglossarystyle{superragged}%

```

Put the glossary in a supertabular environment with two columns, a header and horizontal lines above and below the table:

```

9029 \renewenvironment{theglossary}%
9030 {\tablehead{\hline\bfseries \entryname &
9031 \bfseries \descriptionname\tabularnewline\hline}%
9032 \tabletail{\hline}
9033 \begin{supertabular}{l|>{\raggedright}p{\glsgdescwidth}|}%
9034 {\end{supertabular}}}%
9035 }

```

superragged3col The superragged3col style is like the superragged style, but with 3 columns:

```

9036 \newglossarystyle{superragged3col}{%

```

Put the glossary in a supertabular environment with three columns and no head or tail:

```

9037 \renewenvironment{theglossary}%
9038 {\tablehead{}\tabletail{}}%
9039 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}%
9040 >{\raggedright}p{\glspagelistwidth}}}%
9041 {\end{supertabular}}%

```

Do nothing at the start of the table:

```

9042 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

9043 \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```

9044 \renewcommand{\glossentry}[2]{%
9045 \glstentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9046 \glossentrydesc{##1} &
9047 ##2\tabularnewline
9048 }%

```

Sub entries on a row (no name, description in second column, page list in last column):

```

9049 \renewcommand{\subglossentry}[3]{%
9050 &
9051 \glssubentryitem{##2}%
9052 \glstarget{##2}{\strut}\glossentrydesc{##2} &
9053 ##3\tabularnewline
9054 }%

```

Blank row between groups: The check for nogroupskip must occur outside \glsgroupskip
(<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```
9055 \ifglsgroupskip
9056 \renewcommand*{\glsgroupskip}{}%
9057 \else
9058 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9059 \fi
9060 }
```

agged3colborder The superragged3colborder style is like the superragged3col style, but with a border:

```
9061 \newglossarystyle{superragged3colborder}{%
```

Base it on the glostypesuperragged3col style:

```
9062 \setglossarystyle{superragged3col}%
```

Put the glossary in a supertabular environment with three columns and a horizontal line in the head and tail:

```
9063 \renewenvironment{theglossary}%
9064 {\tablehead{\hline}\tabletail{\hline}%
9065 \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|}%
9066 >{\raggedright}p{\glspagelistwidth}|}%
9067 {\end{supertabular}}%
9068 }
```

agged3colheader The superragged3colheader style is like the superragged3col style but with a header row:

```
9069 \newglossarystyle{superragged3colheader}{%
```

Base it on the glostypesuperragged3col style:

```
9070 \setglossarystyle{superragged3col}%
```

Put the glossary in a supertabular environment with three columns, a header and no tail:

```
9071 \renewenvironment{theglossary}%
9072 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9073 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9074 \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}%
9075 >{\raggedright}p{\glspagelistwidth}}}%
9076 {\end{supertabular}}%
9077 }
```

colheaderborder The superragged3colheaderborder style is like the superragged3col style but with a header and border:

```
9078 \newglossarystyle{superragged3colheaderborder}{%
```

Base it on the glostypesuperragged3colborder style:

```
9079 \setglossarystyle{superragged3colborder}%
```

Put the glossary in a supertabular environment with three columns, a header with horizontal lines and a horizontal line in the tail:

```
9080 \renewenvironment{theglossary}%
9081 {\tablehead{\hline
```

```

9082      \bfseries\entryname&\bfseries\descriptionname&
9083      \bfseries\pagelistname\tabularnewline\hline}%
9084      \tabletail{\hline}%
9085      \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}|%
9086      >{\raggedright}p{\glspagelistwidth}|}%
9087      {\end{supertabular}}%
9088 }

```

superragged4col The altsuperragged4col glossary style is like altsuper4col style in the package but uses ragged right formatting in the description and page list columns.

```

9089 \newglossarystyle{altsuperragged4col}{%

```

Put the glossary in a supertabular environment with four columns and no head or tail:

```

9090 \renewenvironment{theglossary}%
9091 {\tablehead{\tabletail}{}%
9092 \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}l%
9093 >{\raggedright}p{\glspagelistwidth}}}%
9094 {\end{supertabular}}%

```

Do nothing at the start of the table:

```

9095 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

9096 \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries on a row with the name in the first column, description in second column, symbol in third column and page list in last column:

```

9097 \renewcommand{\glossentry}[2]{%
9098 \glssentryitem{##1}\glstarget{##1}{\glossentryname{##1}} &
9099 \glossentrydesc{##1} &
9100 \glossentrysymbol{##1} & ##2\tabularnewline
9101 }%

```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```

9102 \renewcommand{\subglossentry}[3]{%
9103 &
9104 \glssubentryitem{##2}%
9105 \glstarget{##2}{\strut}\glossentrydesc{##2} &
9106 \glossentrysymbol{##2} & ##3\tabularnewline
9107 }%

```

Blank row between groups: The check for nogroupskip must occur outside \glsgroupskip (<http://www.dickimaw-books.com/cgi-bin/bugtracker.cgi?action=view&key=108>)

```

9108 \ifglsgroupskip
9109 \renewcommand*{\glsgroupskip}{}%
9110 \else
9111 \renewcommand*{\glsgroupskip}{& & \tabularnewline}%
9112 \fi
9113 }

```

agged4colheader The altsuperragged4colheader style is like the altsuperragged4col style but with a header row.

```
9114 \newglossarystyle{altsuperragged4colheader}{%
```

Base it on the glostylealtsuperragged4col style:

```
9115 \setglossarystyle{altsuperragged4col}{%
```

Put the glossary in a supertabular environment with four columns, a header and no tail:

```
9116 \renewenvironment{theglossary}{%
9117 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
9118 \bfseries\symbolname &
9119 \bfseries\pagelistname\tabularnewline}\tabletail{}}%
9120 \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}l%
9121 >{\raggedright}p{\glspagelistwidth}}}%
9122 {\end{supertabular}}}%
9123 }
```

agged4colborder The altsuperragged4colborder style is like the altsuperragged4col style but with a border.

```
9124 \newglossarystyle{altsuperragged4colborder}{%
```

Base it on the glostylealtsuperragged4col style:

```
9125 \setglossarystyle{altsuper4col}{%
```

Put the glossary in a supertabular environment with four columns and a horizontal line in the head and tail:

```
9126 \renewenvironment{theglossary}{%
9127 {\tablehead{\hline}\tabletail{\hline}%
9128 \begin{supertabular}%
9129 {ll>{\raggedright}p{\glsgdescwidth}ll}%
9130 >{\raggedright}p{\glspagelistwidth}l}}%
9131 {\end{supertabular}}}%
9132 }
```

colheaderborder The altsuperragged4colheaderborder style is like the altsuperragged4col style but with a header and border.

```
9133 \newglossarystyle{altsuperragged4colheaderborder}{%
```

Base it on the glostylealtsuperragged4col style:

```
9134 \setglossarystyle{altsuperragged4col}{%
```

Put the glossary in a supertabular environment with four columns and a header bordered by horizontal lines and a horizontal line in the tail:

```
9135 \renewenvironment{theglossary}{%
9136 {\tablehead{\hline
9137 \bfseries\entryname &
9138 \bfseries\descriptionname &
9139 \bfseries\symbolname &
9140 \bfseries\pagelistname\tabularnewline\hline}%
9141 \tabletail{\hline}%
9142 \begin{supertabular}%
9143 {ll>{\raggedright}p{\glsgdescwidth}ll}%
9144 >{\raggedright}p{\glspagelistwidth}l}}%
```

```

9145     {\end{supertabular}}}%
9146 }

```

3.10 Tree Styles (glossary-tree.sty)

The style file defines glossary styles that have a tree-like structure. These are designed for hierarchical glossaries.

```

9147 \ProvidesPackage{glossary-tree}[2017/01/19 v4.29 (NLCT)]

```

`\indexspace` There are a few classes that don't define `\indexspace`, so provide a definition if it hasn't been defined.

```

9148 \providecommand{\indexspace}{%
9149   \par \vskip 10\p@ \@plus 5\p@ \@minus 3\p@ \relax
9150 }

```

`\glstreenamefmt` Format used to display the name in the tree styles. (This may be counteracted by `\glslnamefont`.) This command was previously also used to format the group headings.

```

9151 \newcommand*{\glstreenamefmt}[1]{\textbf{#1}}

```

`\glstreegroupheaderfmt` Format used to display the group header in the tree styles. Before v4.22, `\glstreenamefmt` was used for the group header, so the default definition uses that to help maintain backward-compatibility, since in previous versions redefining `\glstreenamefmt` would've also affected the group headings.

```

9152 \newcommand*{\glstreegroupheaderfmt}[1]{\glstreenamefmt{#1}}

```

`\glstreenavigationfmt` Format used to display the navigation header in the tree styles.

```

9153 \newcommand*{\glstreenavigationfmt}[1]{\glstreenamefmt{#1}}

```

Allow the user to adjust the index style without disturbing the index.

`\glstreeitem` Top level item used in index style.

```

9154 \ifdef\@idxitem
9155 {\newcommand{\glstreeitem}{\@idxitem}}
9156 {\newcommand{\glstreeitem}{\par\hangindent40\p@}}

```

`\glstreesubitem` Level 1 item used in index style.

```

9157 \ifdef\subitem
9158 {\let\glstreesubitem\subitem}
9159 {\newcommand\glstreesubitem{\glstreeitem\hspace*{20\p@}}}

```

`\glstreesubsubitem` Level 1 item used in index style.

```

9160 \ifdef\subsubitem
9161 {\let\glstreesubsubitem\subsubitem}
9162 {\newcommand\glstreesubsubitem{\glstreeitem\hspace*{30\p@}}}

```

`\glstreepredesc` Allow the user to adjust the space before the description (except for the `almtree` style).

```

9163 \newcommand{\glstreepredesc}{\space}

```


treechildpredesc Allow the user to adjust the space before the description for sub-entries (except for the treeoname and alttree style).

```
9164 \newcommand{\glstreechildpredesc}{\space}
```

index The index glossary style is similar in style to the way indices are usually typeset using `\item`, `\subitem` and `\subsubitem`. The entry name is set in bold. If an entry has a symbol, it is placed in brackets after the name. Then the description is displayed, followed by the number list. This style allows up to three levels.

```
9165 \newglossarystyle{index}{%
```

Set the paragraph indentation and skip and define `\item` to be the same as that used by theindex:

```
9166 \renewenvironment{theglossary}%
9167 {\setlength{\parindent}{0pt}%
9168 \setlength{\parskip}{0pt plus 0.3pt}%
9169 \let\item\glstreeitem
9170 \let\subitem\glstreesubitem
9171 \let\subsubitem\glstreesubsubitem
9172 }%
```

```
9173 {\par}%
```

Do nothing at the start of the environment:

```
9174 \renewcommand*{\glossaryheader}{}%
```

No group headers:

```
9175 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entry starts a new item with the name in bold followed by the symbol in brackets (if it exists), the description and the page list.

```
9176 \renewcommand*{\glossentry}[2]{%
9177 \item\glssentryitem{##1}\glstreenamfmt{\glstarget{##1}{\glossentryname{##1}}}%
9178 \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9179 \glstreepredesc \glossentrydesc{##1}\glspostdescription\space ##2%
9180 }%
```

Sub entries: level 1 entries use `\subitem`, levels greater than 1 use `\subsubitem`. The level (##1) shouldn't be 0, as that's catered by `\glossentry`, but for completeness, if the level is 0, `\item` is used. The name is put in bold, followed by the symbol in brackets (if it exists), the description and the page list.

```
9181 \renewcommand{\subglossentry}[3]{%
9182 \ifcase##1\relax
9183 % level 0
9184 \item
9185 \or
9186 % level 1
9187 \subitem
9188 \glssubentryitem{##2}%
9189 \else
9190 % all other levels
```

```

9191     \subsubitem
9192     \fi
9193     \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
9194     \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
9195     \glstreechildpredesc\glossentrydesc{##2}\glspostdescription\space ##3%
9196 }%

```

Vertical gap between groups is the same as that used by indices:

```

9197 \renewcommand*{\glsgroupskip}{\ifglsgnogroupskip\else\indexspace\fi}

```

indexgroup The indexgroup style is like the index style but has headings.

```

9198 \newglossarystyle{indexgroup}{%

```

Base it on the glostyleindex style:

```

9199 \setglossarystyle{index}%

```

Add a heading for each group. This puts the group's title in bold followed by a vertical gap.

```

9200 \renewcommand*{\glsgroupheading}[1]{%
9201   \item\glstreegroupheaderfmt{\glsgrouptitle{##1}}%
9202   \indexspace
9203 }%
9204 }

```

indexhypergroup The indexhypergroup style is like the indexgroup style but has hyper navigation.

```

9205 \newglossarystyle{indexhypergroup}{%

```

Base it on the glostyleindex style:

```

9206 \setglossarystyle{index}%

```

Put navigation links to the groups at the start of the glossary:

```

9207 \renewcommand*{\glossaryheader}{%
9208   \item\glstreenavigationfmt{\glsgroupnavigation}\indexspace}%

```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```

9209 \renewcommand*{\glsgroupheading}[1]{%
9210   \item\glstreegroupheaderfmt
9211     {\glsgroupnavigationtarget{##1}{\glsgrouptitle{##1}}}%
9212   \indexspace}%
9213 }

```

tree The tree glossary style is similar in style to the index style, but can have arbitrary levels.

```

9214 \newglossarystyle{tree}{%

```

Set the paragraph indentation and skip:

```

9215 \renewenvironment{theglossary}%
9216   {\setlength{\parindent}{0pt}%
9217    \setlength{\parskip}{0pt plus 0.3pt}}%
9218   {}%

```

Do nothing at the start of the theglossary environment:

```

9219 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```
9220 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries: name in bold, followed by symbol in brackets (if it exists), the description and the page list:

```
9221 \renewcommand{\glossentry}[2]{%
9222   \hangindent0pt\relax
9223   \parindent0pt\relax
9224   \glstentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}%
9225   \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9226   \glstreepredesc\glossentrydesc{##1}\glspostdescription\space##2\par
9227 }%
```

Sub entries: level $\langle n \rangle$ is indented by $\langle n \rangle$ times `\glstreeindent`. The name is in bold, followed by the symbol in brackets (if it exists), the description and the page list.

```
9228 \renewcommand{\subglossentry}[3]{%
9229   \hangindent##1\glstreeindent\relax
9230   \parindent##1\glstreeindent\relax
9231   \ifnum##1=1\relax
9232     \glssubentryitem{##2}%
9233     \fi
9234     \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}%
9235     \ifglshassymbol{##2}{\space(\glossentrysymbol{##2})}{}%
9236     \glstreechildpredesc\glossentrydesc{##2}\glspostdescription\space ##3\par
9237 }%
```

Vertical gap between groups is the same as that used by indices:

```
9238 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}}
```

treegroup Like the tree style but the glossary groups have headings.

```
9239 \newglossarystyle{treegroup}{%
```

Base it on the `glostyletree` style:

```
9240 \setglossarystyle{tree}%
```

Each group has a heading (in bold) followed by a vertical gap):

```
9241 \renewcommand{\glsgroupheading}[1]{\par
9242   \noindent\glstreegroupheaderfmt{\glsgrouptitle{##1}}\par
9243   \indexspace}%
9244 }
```

treehypergroup The `treehypergroup` style is like the `treegroup` style, but has a set of links to the groups at the start of the glossary.

```
9245 \newglossarystyle{treehypergroup}{%
```

Base it on the `glostyletree` style:

```
9246 \setglossarystyle{tree}%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
9247 \renewcommand*{\glossaryheader}{%
9248   \par\noindent\glstreenavigationfmt{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```

9249 \renewcommand*{\glsgroupheading}[1]{%
9250   \par\noindent
9251   \glstreegroupheaderfmt
9252   {\glssnavhypertarget{##1}{\glsgrouptitle{##1}}}\par
9253   \indexspace}%
9254 }
```

`\glstreeindent` Length governing left indent for each level of the tree style.

```

9255 \newlength\glstreeindent
9256 \setlength{\glstreeindent}{10pt}
```

`treenoname` The `treenoname` glossary style is like the tree style, but doesn't print the name or symbol for sub-levels.

```
9257 \newglossarystyle{treenoname}{%
```

Set the paragraph indentation and skip:

```

9258 \renewenvironment{theglossary}%
9259   {\setlength{\parindent}{0pt}%
9260    \setlength{\parskip}{0pt plus 0.3pt}}%
9261   {}%
```

No header:

```
9262 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
9263 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries: the name is in bold, followed by the symbol in brackets (if it exists), the description and the page list.

```

9264 \renewcommand{\glossentry}[2]{%
9265   \hangindent0pt\relax
9266   \parindent0pt\relax
9267   \glssentryitem{##1}\glstreenamfmt{\glstarget{##1}{\glossentryname{##1}}}%
9268   \ifglshassymbol{##1}{\space(\glossentrysymbol{##1})}{}%
9269   \glstreepredesc\glossentrydesc{##1}\glspostdescription\space##2\par
9270   }%
```

Sub entries: level $\langle n \rangle$ is indented by $\langle n \rangle$ times `\glstreeindent`. The name and symbol are omitted. The description followed by the page list are displayed.

```

9271 \renewcommand{\subglossentry}[3]{%
9272   \hangindent##1\glstreeindent\relax
9273   \parindent##1\glstreeindent\relax
9274   \ifnum##1=1\relax
9275     \glssubentryitem{##2}%
9276   \fi
9277   \glstarget{##2}{\strut}%
9278   \glossentrydesc{##2}\glspostdescription\space##3\par
9279   }%
```

Vertical gap between groups is the same as that used by indices:

```
9280 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
9281 }
```

treenonamegroup Like the `treenoname` style but the glossary groups have headings.

```
9282 \newglossarystyle{treenonamegroup}{%
```

Base it on the `glostyletreenoname` style:

```
9283 \setglossarystyle{treenoname}%
```

Give each group a heading:

```
9284 \renewcommand{\glsgroupheading}[1]{\par
9285 \noindent\glstreegroupheaderfmt
9286 {\glsgrouptitle{##1}}\par\indexspace}%
9287 }
```

treenonamehypergroup The `treenonamehypergroup` style is like the `treenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```
9288 \newglossarystyle{treenonamehypergroup}{%
```

Base it on the `glostyletreenoname` style:

```
9289 \setglossarystyle{treenoname}%
```

Put navigation links to the groups at the start of the `theglossary` environment:

```
9290 \renewcommand*{\glossaryheader}{%
9291 \par\noindent\glstreenavigationfmt{\glstravel}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
9292 \renewcommand*{\glsgroupheading}[1]{%
9293 \par\noindent
9294 \glstreegroupheaderfmt
9295 {\glstravelhypertarget{##1}{\glsgrouptitle{##1}}}\par
9296 \indexspace}%
9297 }
```

esttoplevelname Find the widest name over all parentless entries in the given glossary or glossaries.

```
9298 \newrobustcmd*{\glsfindwidesttoplevelname}[1][\@glo@types]{%
9299 \dimen@=0pt\relax
9300 \gls@tmplen=0pt\relax
9301 \forallglossaries[#1]{\@gls@type}%
9302 {%
9303 \forallgsentries[\@gls@type]{\@glo@label}%
9304 {%
9305 \ifglshasparent{\@glo@label}%
9306 }%
9307 {%
9308 \settowidth{\dimen@}%
9309 {\glstreenamelfmt{\glsentryname{\@glo@label}}}%
9310 \ifdim\dimen@>\gls@tmplen
9311 \gls@tmplen=\dimen@
```

```

9312         \letcs{\@glswidestname}{glo@\glsdetoklabel{\@glo@label}@name}%
9313         \fi
9314     }%
9315 }%
9316 }%
9317 }

```

`\glsetwidest` `\glsetwidest[<level>]{<text>}` sets the widest text for the given level. It is used by the alt-tree glossary styles to determine the indentation of each level.

```

9318 \newcommand*{\glsetwidest}[2][0]{%
9319     \expandafter\def\csname @glswidestname\romannumeral#1\endcsname{%
9320         #2}%
9321 }

```

`\@glswidestname` Initialise `\@glswidestname`.

```

9322 \newcommand*{\@glswidestname}{}

```

`\glstreenamebox` Used by the alttree style to create the box for the name and associated information.

```

9323 \newcommand*{\glstreenamebox}[2]{%
9324     \makebox[#1][l]{#2}%
9325 }

```

alttree The alttree glossary style is similar in style to the tree style, but the indentation is obtained from the width of `\@glswidestname` which is set using `\glsetwidest`.

```

9326 \newglossarystyle{alttree}{%

```

Redefine theglossary environment.

```

9327     \renewenvironment{theglossary}%
9328     {\def\@gls@prevlevel{-1}%
9329     \mbox{}\par}%
9330     {\par}%

```

Set the header and group headers to nothing.

```

9331     \renewcommand*{\glossaryheader}{}%
9332     \renewcommand*{\glsgroupheading}[1]{}%

```

Redefine the way that the level 0 entries are displayed.

```

9333     \renewcommand{\glossentry}[2]{%
9334         \ifnum\@gls@prevlevel=0\relax
9335         \else

```

Find out how big the indentation should be by measuring the widest entry.

```

9336         \settowidth{\glstreeindent}{\glstreenamefmt{\@glswidestname\space}}%
9337         \fi

```

Set the hangindent and paragraph indent.

```

9338         \hangindent\glstreeindent
9339         \parindent\glstreeindent

```

Put the name to the left of the paragraph block.

```

9340         \makebox[0pt][r]{\glstreenamebox{\glstreeindent}}%
9341         \glsentryitem{##1}\glstreenamefmt{\glstarget{##1}{\glossentryname{##1}}}}}%

```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
9342 \ifglshassymbol{##1}{(\glossentrysymbol{##1})\space}{}%
```

Do the description followed by the description terminator and location list.

```
9343 \glossentrydesc{##1}\glspostdescription \space ##2\par
```

Set the previous level to 0.

```
9344 \def\@gls@prevlevel{0}%
```

```
9345 }%
```

Redefine the way sub-entries are displayed.

```
9346 \renewcommand{\subglossentry}[3]{%
```

Increment and display the sub-entry counter if this is a level 1 entry and the sub-entry counter is in use.

```
9347 \ifnum##1=1\relax
```

```
9348 \glssubentryitem{##2}%
```

```
9349 \fi
```

If the level hasn't changed, keep the same settings, otherwise adjust `\glstreeindent` accordingly.

```
9350 \ifnum\@gls@prevlevel=##1\relax
```

```
9351 \else
```

Compute the widest entry for this level, or for level 0 if not defined for this level. Store in `\gls@tmplen`

```
9352 \@ifundefined{@glswidestname\romannumeral##1}{%
```

```
9353 \settowidth{\gls@tmplen}{\glstreenamefmt{\@glswidestname\space}}{%
```

```
9354 \settowidth{\gls@tmplen}{\glstreenamefmt{%
```

```
9355 \csname @glswidestname\romannumeral##1\endcsname\space}}{%
```

Determine if going up or down a level

```
9356 \ifnum\@gls@prevlevel<##1\relax
```

Depth has increased, so add the width of the widest entry to `\glstreeindent`.

```
9357 \setlength\glstreeindent\gls@tmplen
```

```
9358 \addtolength\glstreeindent\parindent
```

```
9359 \parindent\glstreeindent
```

```
9360 \else
```

Depth has decreased, so subtract width of the widest entry from the previous level to `\glstreeindent`. First determine the width of the widest entry for the previous level and store in `\glstreeindent`.

```
9361 \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
```

```
9362 \settowidth{\glstreeindent}{\glstreenamefmt{%
```

```
9363 \@glswidestname\space}}{%
```

```
9364 \settowidth{\glstreeindent}{\glstreenamefmt{%
```

```
9365 \csname @glswidestname\romannumeral\@gls@prevlevel
```

```
9366 \endcsname\space}}{%
```

Subtract this length from the previous level's paragraph indent and set to `\glstreeindent`.

```
9367 \addtolength\parindent{-\glstreeindent}%
```

```

9368         \setlength\glstreeindent\parindent
9369     \fi
9370 \fi

    Set the hanging indentation.
9371     \hangindent\glstreeindent

    Put the name to the left of the paragraph block
9372     \makebox[0pt][r]{\glstreenamebox{\gls@tmplen}{%
9373         \glstreenamefmt{\glstarget{##2}{\glossentryname{##2}}}}}%

    If the symbol is missing, ignore it, otherwise put it in brackets.
9374     \ifglshassymbol{##2}{(\glossentrysymbol{##2})\space}{}%

    Do the description followed by the description terminator and location list.
9375     \glossentrydesc{##2}\glspostdescription\space ##3\par

    Set the previous level macro to the current level.
9376     \def\@gls@prevlevel{##1}%
9377 }%

    Vertical gap between groups is the same as that used by indices:
9378 \renewcommand*{\glsgroupskip}{\ifglsnogroupskip\else\indexspace\fi}%
9379 }

```

alttreegroup Like the **alttree** style but the glossary groups have headings.

```

9380 \newglossarystyle{alttreegroup}{%
    Base it on the glostylealttree style:
9381     \setglossarystyle{alttree}%

    Give each group a heading.
9382     \renewcommand{\glsgroupheading}[1]{\par
9383         \def\@gls@prevlevel{-1}%
9384         \hangindent0pt\relax
9385         \parindent0pt\relax
9386         \glstreegroupheaderfmt{\glsgetgrouptitle{##1}}%
9387         \par\indexspace}%
9388 }

```

alttreehypergroup The **alttreehypergroup** style is like the **alttreegroup** style, but has a set of links to the groups at the start of the glossary.

```

9389 \newglossarystyle{alttreehypergroup}{%
    Base it on the glostylealttree style:
9390     \setglossarystyle{alttree}%

    Put the navigation links in the header
9391     \renewcommand*{\glossaryheader}{%
9392         \par
9393         \def\@gls@prevlevel{-1}%
9394         \hangindent0pt\relax
9395         \parindent0pt\relax
9396         \glstreenavigationfmt{\glsnavigation}\par\indexspace}%

```


Put a hypertarget at the start of each group

```
9397 \renewcommand*{\glsgroupheading}[1]{%
9398   \par
9399   \def\@gls@prevlevel{-1}%
9400   \hangindent0pt\relax
9401   \parindent0pt\relax
9402   \glstreegroupheaderfmt
9403   {\@glsnavhypertarget{##1}{\@glsgetgrouptitle{##1}}}\par
9404   \indexspace}}
```

4 Backwards Compatibility

4.1 glossaries-compatible-207

Provides compatibility with version 2.07 and below. This uses original glossaries xindy and makeindex formatting, so can be used with old documents that had customized style files, but hyperlinks may not work properly.

```
9405 \NeedsTeXFormat{LaTeX2e}
9406 \ProvidesPackage{glossaries-compatible-207}[2017/01/19 v4.29 (NLCT)]
```

AddXdyAttribute Adds an attribute in old format.

```
9407 \ifglxsindy
9408   \renewcommand*\GlsAddXdyAttribute[1]{%
9409     \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string"}%
9410     \expandafter\toks@\expandafter{\@xdylocref}%
9411     \edef\@xdylocref{\the\toks@ ^^J%
9412     (markup-locref
9413     :open \string"\string~n\string\setentrycounter
9414     {\noexpand\glscounter}%
9415     \expandafter\string\csname#1\endcsname
9416     \expandafter\@gobble\string\{\string" ^^J
9417     :close \string"\expandafter\@gobble\string\}\string" ^^J
9418     :attr \string"#1\string")}}
```

Only has an effect before `\writeist`:

```
9419 \fi
```

sAddXdyCounters

```
9420 \renewcommand*\GlsAddXdyCounters[1]{%
9421   \GlossariesWarning{\string\GlsAddXdyCounters\space not available
9422     in compatibility mode.}%
9423 }
```

Add predefined attributes

```
9424 \GlsAddXdyAttribute{glsnumberformat}
9425 \GlsAddXdyAttribute{textrm}
9426 \GlsAddXdyAttribute{textsf}
9427 \GlsAddXdyAttribute{texttt}
9428 \GlsAddXdyAttribute{textbf}
9429 \GlsAddXdyAttribute{textmd}
9430 \GlsAddXdyAttribute{textit}
9431 \GlsAddXdyAttribute{textup}
9432 \GlsAddXdyAttribute{textsl}
```

```

9433 \GlsAddXdyAttribute{textsc}
9434 \GlsAddXdyAttribute{emph}
9435 \GlsAddXdyAttribute{glshypernumber}
9436 \GlsAddXdyAttribute{hyperrm}
9437 \GlsAddXdyAttribute{hypersf}
9438 \GlsAddXdyAttribute{hypertt}
9439 \GlsAddXdyAttribute{hyperbf}
9440 \GlsAddXdyAttribute{hypermd}
9441 \GlsAddXdyAttribute{hyperit}
9442 \GlsAddXdyAttribute{hyperup}
9443 \GlsAddXdyAttribute{hypersl}
9444 \GlsAddXdyAttribute{hypersc}
9445 \GlsAddXdyAttribute{hyperemph}

```

sAddXdyLocation Restore v2.07 definition:

```

9446 \ifglxindy
9447 \renewcommand*{\GlsAddXdyLocation}[2]{%
9448   \edef\@xdyuserlocationdefs{%
9449     \@xdyuserlocationdefs ^^J%
9450     (define-location-class \string"#1\string"^^J\space\space
9451     \space(#2))
9452   }%
9453   \edef\@xdyuserlocationnames{%
9454     \@xdyuserlocationnames^^J\space\space\space
9455     \string"#1\string"}%
9456   }
9457 \fi

```

\@do@wrglossary

```

9458 \renewcommand{\@do@wrglossary}[1]{%
  Determine whether to use xindy or makeindex syntax
9459 \ifglxindy
  Need to determine if the formatting information starts with a ( or ) indicating a range.
9460 \expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil
9461 \def\@glo@range{}%
9462 \expandafter\if\@glo@prefix(\relax
9463   \def\@glo@range{:open-range}%
9464 \else
9465   \expandafter\if\@glo@prefix)\relax
9466   \def\@glo@range{:close-range}%
9467 \fi
9468 \fi

  Get the location and escape any special characters
9469 \protected@edef\@glslocref{\theglsentrycounter}%
9470 \@gls@checkmkidxchars\@glslocref

  Write to the glossary file using xindy syntax.
9471 \glossary[\csname glo@#1@type\endcsname]{%

```

```

9472 (indexentry :tkey (\csname glo@#1@index\endcsname)
9473   :locoref \string"\@glslocoref\string" %
9474   :attr \string"\@glo@suffix\string" \@glo@range
9475 )
9476 }%
9477 \else
    Convert the format information into the format required for makeindex
9478   \@set@glo@numformat\@glo@numfmt\@gls@counter\@glsnumberformat
    Write to the glossary file using makeindex syntax.
9479   \glossary[\csname glo@#1@type\endcsname]{%
9480     \string\glossaryentry{\csname glo@#1@index\endcsname
9481       \@gls@encapchar\@glo@numfmt}{\theglsentrycounter}}%
9482 \fi
9483 }

```

t@glo@numformat Only had 3 arguments in v2.07

```

9484 \def\@set@glo@numformat#1#2#3{%
9485   \expandafter\@glo@check@mkidxrangechar#3\@nil
9486   \protected@edef#1{%
9487     \@glo@prefix setentrycounter[] {#2}%
9488     \expandafter\string\csname\@glo@suffix\endcsname
9489   }%
9490   \@gls@checkmkidxchars#1%
9491 }

```

\writeist Redefine \writeist back to the way it was in v2.07, but change \istfile to \glswrite.

```

9492 \ifglxsindy
9493   \def\writeist{%
9494     \openout\glswrite=\istfilename
9495     \write\glswrite{;; xindy style file created by the glossaries
9496       package in compatible-2.07 mode}%
9497     \write\glswrite{;; for document '\jobname' on
9498       \the\year-\the\month-\the\day}%
9499     \write\glswrite{^^J; required styles^^J}
9500     \@for\@xdystyle:=\@xdyrequiredstyles\do{%
9501       \ifx\@xdystyle\@empty
9502       \else
9503         \protected@write\glswrite{{(require
9504           \string"\@xdystyle.xdy\string")}}%
9505       \fi
9506     }%
9507     \write\glswrite{^^J%
9508       ; list of allowed attributes (number formats)^^J}%
9509     \write\glswrite{(define-attributes ((\@xdyattributes)))}%
9510     \write\glswrite{^^J; user defined alphabets^^J}%
9511     \write\glswrite{\@xdyuseralphabets}%
9512     \write\glswrite{^^J; location class definitions^^J}%
9513     \protected@edef\@gls@roman{\@roman{0}\string"

```

```

9514     \string"roman-numbers-lowercase\string" :sep \string"}}%
9515 \@onelevel@sanitize\@gls@roman
9516 \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
9517     :sep \string"}}%
9518 \@onelevel@sanitize\@tmp
9519 \ifx\@tmp\@gls@roman
9520     \write\glswrite{(define-location-class
9521         \string"roman-page-numbers\string"^^J\space\space\space
9522         (\string"roman-numbers-lowercase\string")
9523         :min-range-length \@glsminrange)}}%
9524 \else
9525     \write\glswrite{(define-location-class
9526         \string"roman-page-numbers\string"^^J\space\space\space
9527         (:sep "\@gls@roman")
9528         :min-range-length \@glsminrange)}}%
9529 \fi
9530 \write\glswrite{(define-location-class
9531     \string"Roman-page-numbers\string"^^J\space\space\space
9532     (\string"roman-numbers-uppercase\string")
9533     :min-range-length \@glsminrange)}}%
9534 \write\glswrite{(define-location-class
9535     \string"arabic-page-numbers\string"^^J\space\space\space
9536     (\string"arabic-numbers\string")
9537     :min-range-length \@glsminrange)}}%
9538 \write\glswrite{(define-location-class
9539     \string"alpha-page-numbers\string"^^J\space\space\space
9540     (\string"alpha\string")
9541     :min-range-length \@glsminrange)}}%
9542 \write\glswrite{(define-location-class
9543     \string"Alpha-page-numbers\string"^^J\space\space\space
9544     (\string"ALPHA\string")
9545     :min-range-length \@glsminrange)}}%
9546 \write\glswrite{(define-location-class
9547     \string"Appendix-page-numbers\string"^^J\space\space\space
9548     (\string"ALPHA\string"
9549     :sep \string"\@glsAlphacompositor\string"
9550     \string"arabic-numbers\string")
9551     :min-range-length \@glsminrange)}}%
9552 \write\glswrite{(define-location-class
9553     \string"arabic-section-numbers\string"^^J\space\space\space
9554     (\string"arabic-numbers\string"
9555     :sep \string"\glscompositor\string"
9556     \string"arabic-numbers\string")
9557     :min-range-length \@glsminrange)}}%
9558 \write\glswrite{^^J; user defined location classes}%
9559 \write\glswrite{\@xdyuserlocationdefs}%
9560 \write\glswrite{^^J; define cross-reference class^^J}%
9561 \write\glswrite{(define-crossref-class \string"see\string"
9562     :unverified )}%

```

```

9563 \write\glswrite{(markup-crossref-list
9564 :class \string"see\string"^^J\space\space\space
9565 :open \string"\string\glseeformat\string"
9566 :close \string"{}\string")}%
9567 \write\glswrite{^^J; define the order of the location classes}%
9568 \write\glswrite{(define-location-class-order
9569 (\@xdylocationclassorder))}%
9570 \write\glswrite{^^J; define the glossary markup^^J}%
9571 \write\glswrite{(markup-index^^J\space\space\space
9572 :open \string"\string
9573 \glossarysection[\string\glossarytoctitle]{\string
9574 \glossarytitle}\string\glossarypreamble\string~n\string\begin
9575 {theglossary}\string\glossaryheader\string~n\string" ^^J\space
9576 \space\space:close \string"\expandafter\@gobble
9577 \string%\string~n\string
9578 \end{theglossary}\string\glossarypostamble
9579 \string~n\string" ^^J\space\space\space
9580 :tree)}}%
9581 \write\glswrite{(markup-letter-group-list
9582 :sep \string"\string\glsgroupskip\string~n\string")}%
9583 \write\glswrite{(markup-indexentry
9584 :open \string"\string\relax \string\glresetentrylist
9585 \string~n\string")}%
9586 \write\glswrite{(markup-locclass-list :open
9587 \string"\glsoopenbrace\string\glossaryentrynumbers
9588 \glsoopenbrace\string\relax\space \string"^^J\space\space\space
9589 :sep \string", \string"
9590 :close \string"\glsclosebrace\glsclosebrace\string")}%
9591 \write\glswrite{(markup-locref-list
9592 :sep \string"\string\delimN\space\string")}%
9593 \write\glswrite{(markup-range
9594 :sep \string"\string\delimR\space\string")}%
9595 \@onelevel@sanitize\gls@suffixF
9596 \@onelevel@sanitize\gls@suffixFF
9597 \ifx\gls@suffixF\@empty
9598 \else
9599 \write\glswrite{(markup-range
9600 :close "\gls@suffixF" :length 1 :ignore-end)}%
9601 \fi
9602 \ifx\gls@suffixFF\@empty
9603 \else
9604 \write\glswrite{(markup-range
9605 :close "\gls@suffixFF" :length 2 :ignore-end)}%
9606 \fi
9607 \write\glswrite{^^J; define format to use for locations^^J}%
9608 \write\glswrite{\@xdylocref}%
9609 \write\glswrite{^^J; define letter group list format^^J}%
9610 \write\glswrite{(markup-letter-group-list
9611 :sep \string"\string\glsgroupskip\string~n\string")}%

```

```

9612 \write\glswrite{^^J; letter group headings^^J}%
9613 \write\glswrite{(markup-letter-group
9614   :open-head \string"\string\glsgroupheading
9615   \glsoopenbrace\string"^^J\space\space\space
9616   :close-head \string"\glsclosebrace\string")}%
9617 \write\glswrite{^^J; additional letter groups^^J}%
9618 \write\glswrite{\@xdylettergroups}%
9619 \write\glswrite{^^J; additional sort rules^^J}
9620 \write\glswrite{\@xdysortrules}%
9621 \noist}
9622 \else
9623 \edef\@gls@actualchar{\string?}
9624 \edef\@gls@encapchar{\string|}
9625 \edef\@gls@levelchar{\string!}
9626 \edef\@gls@quotechar{\string"}
9627 \def\writeist{\relax
9628   \openout\glswrite=\istfilename
9629   \write\glswrite{\expandafter\@gobble\string\% makeindex style file
9630     created by the glossaries package}
9631   \write\glswrite{\expandafter\@gobble\string\% for document
9632     '\jobname' on \the\year-\the\month-\the\day}
9633   \write\glswrite{actual '\@gls@actualchar'}
9634   \write\glswrite{encap '\@gls@encapchar'}
9635   \write\glswrite{level '\@gls@levelchar'}
9636   \write\glswrite{quote '\@gls@quotechar'}
9637   \write\glswrite{keyword \string"\string\glossaryentry\string"}
9638   \write\glswrite{preamble \string"\string\glossarysection[\string
9639     \glossarytoctitle]{\string\glossarytitle}\string
9640     \glossarypreamble\string\n\string\begin{theglossary}\string
9641     \glossaryheader\string\n\string"}
9642   \write\glswrite{postamble \string"\string%\string\n\string
9643     \end{theglossary}\string\glossarypostamble\string\n
9644     \string"}
9645   \write\glswrite{group_skip \string"\string\glsgroupskip\string\n
9646     \string"}
9647   \write\glswrite{item_0 \string"\string%\string\n\string"}
9648   \write\glswrite{item_1 \string"\string%\string\n\string"}
9649   \write\glswrite{item_2 \string"\string%\string\n\string"}
9650   \write\glswrite{item_01 \string"\string%\string\n\string"}
9651   \write\glswrite{item_x1
9652     \string"\string\relax \string\glsresetentrylist\string\n
9653     \string"}
9654   \write\glswrite{item_12 \string"\string%\string\n\string"}
9655   \write\glswrite{item_x2
9656     \string"\string\relax \string\glsresetentrylist\string\n
9657     \string"}
9658   \write\glswrite{delim_0 \string"\string\{\string
9659     \glossaryentrynumbers\string\{\string\relax \string"}
9660   \write\glswrite{delim_1 \string"\string\{\string

```

```

9661      \glossaryentrynumbers\string\{\string\relax \string}
9662      \write\glswrite{delim_2 \string"\string\{\string
9663      \glossaryentrynumbers\string\{\string\relax \string}
9664      \write\glswrite{delim_t \string"\string\}\string\}\string}
9665      \write\glswrite{delim_n \string"\string\delimN \string}
9666      \write\glswrite{delim_r \string"\string\delimR \string}
9667      \write\glswrite{headings_flag 1}
9668      \write\glswrite{heading_prefix
9669          \string"\string\glsgroupheading\string\{\string}
9670      \write\glswrite{heading_suffix
9671          \string"\string\}\string\relax
9672          \string\glsgroupresetentrylist \string}
9673      \write\glswrite{symhead_positive \string"glssymbols\string}
9674      \write\glswrite{numhead_positive \string"glslnumbers\string}
9675      \write\glswrite{page_compositor \string"glscpositor\string}
9676      \@gls@escbsdq\gls@suffixF
9677      \@gls@escbsdq\gls@suffixFF
9678      \ifx\gls@suffixF\@empty
9679      \else
9680          \write\glswrite{suffix_2p \string"\gls@suffixF\string}
9681      \fi
9682      \ifx\gls@suffixFF\@empty
9683      \else
9684          \write\glswrite{suffix_3p \string"\gls@suffixFF\string}
9685      \fi
9686      \noist
9687  }
9688 \fi

```

\noist

```

9689 \renewcommand*{\noist}{\let\writeist\relax}

```

4.2 glossaries-compatible-307

```

9690 \NeedsTeXFormat{LaTeX2e}
9691 \ProvidesPackage{glossaries-compatible-307}[2017/01/19 v4.29 (NLCT)]

```

Compatibility macros for predefined glossary styles:

`\atglossarystyle` Defines a compatibility glossary style.

```

9692 \newcommand{\compatglossarystyle}[2]{%
9693   \ifcsundef{@glscompstyle@#1}%
9694   {%
9695     \csdef{@glscompstyle@#1}{#2}%
9696   }%
9697   {%
9698     \PackageError{glossaries}{Glossary compatibility style ‘#1’ is already defined}{}%
9699   }%
9700 }

```


Backward compatible inline style.

```

9701 \compatglossarystyle{inline}{%
9702   \renewcommand{\glossaryentryfield}[5]{%
9703     \glsinlinedopostchild
9704     \gls@inlinesep
9705     \def\glo@desc{##3}%
9706     \def\@no@post@desc{\nopostdesc}%
9707     \glstentryitem{##1}\glsinlinenameformat{##1}{##2}%
9708     \ifx\glo@desc\@no@post@desc
9709       \glsinlineemptydescformat{##4}{##5}%
9710     \else
9711       \ifstrepty{##3}%
9712         {\glsinlineemptydescformat{##4}{##5}}%
9713         {\glsinlinedescformat{##3}{##4}{##5}}%
9714     \fi
9715     \ifglshaschildren{##1}%
9716     {%
9717       \glsresetsubentrycounter
9718       \glsinlineparentchildseparator
9719       \def\gls@inlinesubsep{}%
9720       \def\gls@inlinepostchild{\glsinlinepostchild}%
9721     }%
9722   }%
9723   \def\gls@inlinesep{\glsinlineseparator}%
9724 }%
```

Sub-entries display description:

```

9725 \renewcommand{\glossarysubentryfield}[6]{%
9726   \gls@inlinesubsep%
9727   \glsinlinesubnameformat{##2}{##3}%
9728   \glssubentryitem{##2}\glsinlinesubdescformat{##4}{##5}{##6}%
9729   \def\gls@inlinesubsep{\glsinlinesubseparator}%
9730 }%
9731 }
```

Backward compatible list style.

```

9732 \compatglossarystyle{list}{%
9733   \renewcommand*{\glossaryentryfield}[5]{%
9734     \item[\glstentryitem{##1}\glstarget{##1}{##2}]
9735     ##3\glspostdescription\space ##5}%
9736 }
```

Sub-entries continue on the same line:

```

9736 \renewcommand*{\glossarysubentryfield}[6]{%
9737   \glssubentryitem{##2}%
9738   \glstarget{##2}{\strut}##4\glspostdescription\space ##6.}%
9739 }
```

Backward compatible listgroup style.

```

9740 \compatglossarystyle{listgroup}{%
9741   \csuse{@glscompstyle@list}%
9742 }%
```

Backward compatible listhypergroup style.

```
9743 \compatglossarystyle{listhypergroup}{%
9744   \csuse{@glscompstyle@list}%
9745 }%
```

Backward compatible altlist style.

```
9746 \compatglossarystyle{altlist}{%
9747   \renewcommand*{\glossaryentryfield}[5]{%
9748     \item[\glstentryitem{##1}\glstarget{##1}{##2}]%
9749     \mbox{}\par\nobreak\@afterheading
9750     ##3\glspostdescription\space ##5}%
9751   \renewcommand{\glossarysubentryfield}[6]{%
9752     \par
9753     \glssubentryitem{##2}%
9754     \glstarget{##2}{\strut}##4\glspostdescription\space ##6}%
9755 }%
```

Backward compatible altlistgroup style.

```
9756 \compatglossarystyle{altlistgroup}{%
9757   \csuse{@glscompstyle@altlist}%
9758 }%
```

Backward compatible altlisthypergroup style.

```
9759 \compatglossarystyle{altlisthypergroup}{%
9760   \csuse{@glscompstyle@altlist}%
9761 }%
```

Backward compatible listdotted style.

```
9762 \compatglossarystyle{listdotted}{%
9763   \renewcommand*{\glossaryentryfield}[5]{%
9764     \item[\makebox[\glslistdottedwidth][l]{%
9765       \glstentryitem{##1}\glstarget{##1}{##2}%
9766       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##3}%
9767   \renewcommand*{\glossarysubentryfield}[6]{%
9768     \item[\makebox[\glslistdottedwidth][l]{%
9769       \glssubentryitem{##2}%
9770       \glstarget{##2}{##3}%
9771       \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##4}%
9772 }%
```

Backward compatible sublistdotted style.

```
9773 \compatglossarystyle{sublistdotted}{%
9774   \csuse{@glscompstyle@listdotted}%
9775   \renewcommand*{\glossaryentryfield}[5]{%
9776     \item[\glstentryitem{##1}\glstarget{##1}{##2}]}%
9777 }%
```

Backward compatible long style.

```
9778 \compatglossarystyle{long}{%
9779   \renewcommand*{\glossaryentryfield}[5]{%
9780     \glstentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
9781   \renewcommand*{\glossarysubentryfield}[6]{%

```

```

9782      &
9783      \glssubentryitem{##2}%
9784      \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
9785 }%

```

Backward compatible longborder style.

```

9786 \compatglossarystyle{longborder}{%
9787 \csuse{@glscmpstyle@long}%
9788 }%

```

Backward compatible longheader style.

```

9789 \compatglossarystyle{longheader}{%
9790 \csuse{@glscmpstyle@long}%
9791 }%

```

Backward compatible longheaderborder style.

```

9792 \compatglossarystyle{longheaderborder}{%
9793 \csuse{@glscmpstyle@long}%
9794 }%

```

Backward compatible long3col style.

```

9795 \compatglossarystyle{long3col}{%
9796 \renewcommand*{\glossaryentryfield}[5]{%
9797 \glstarget{##1}{\glstarget{##1}{##2} & ##3 & ##5\\}%
9798 \renewcommand*{\glossarysubentryfield}[6]{%
9799 &
9800 \glssubentryitem{##2}%
9801 \glstarget{##2}{\strut}##4 & ##6\\}%
9802 }%

```

Backward compatible long3colborder style.

```

9803 \compatglossarystyle{long3colborder}{%
9804 \csuse{@glscmpstyle@long3col}%
9805 }%

```

Backward compatible long3colheader style.

```

9806 \compatglossarystyle{long3colheader}{%
9807 \csuse{@glscmpstyle@long3col}%
9808 }%

```

Backward compatible long3colheaderborder style.

```

9809 \compatglossarystyle{long3colheaderborder}{%
9810 \csuse{@glscmpstyle@long3col}%
9811 }%

```

Backward compatible long4col style.

```

9812 \compatglossarystyle{long4col}{%
9813 \renewcommand*{\glossaryentryfield}[5]{%
9814 \glstarget{##1}{\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
9815 \renewcommand*{\glossarysubentryfield}[6]{%
9816 &
9817 \glssubentryitem{##2}%

```

9818 \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
 9819 }%

Backward compatible long4colheader style.

9820 \compatglossarystyle{long4colheader}{%
 9821 \csuse{@glscompstyle@long4col}%
 9822 }%

Backward compatible long4colborder style.

9823 \compatglossarystyle{long4colborder}{%
 9824 \csuse{@glscompstyle@long4col}%
 9825 }%

Backward compatible long4colheaderborder style.

9826 \compatglossarystyle{long4colheaderborder}{%
 9827 \csuse{@glscompstyle@long4col}%
 9828 }%

Backward compatible altlong4col style.

9829 \compatglossarystyle{altlong4col}{%
 9830 \csuse{@glscompstyle@long4col}%
 9831 }%

Backward compatible altlong4colheader style.

9832 \compatglossarystyle{altlong4colheader}{%
 9833 \csuse{@glscompstyle@long4col}%
 9834 }%

Backward compatible altlong4colborder style.

9835 \compatglossarystyle{altlong4colborder}{%
 9836 \csuse{@glscompstyle@long4col}%
 9837 }%

Backward compatible altlong4colheaderborder style.

9838 \compatglossarystyle{altlong4colheaderborder}{%
 9839 \csuse{@glscompstyle@long4col}%
 9840 }%

Backward compatible long style.

9841 \compatglossarystyle{longragged}{%
 9842 \renewcommand*{\glossaryentryfield}[5]{%
 9843 \glssentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
 9844 \tabularnewline}%
 9845 \renewcommand*{\glossarysubentryfield}[6]{%
 9846 &
 9847 \glssubentryitem{##2}%
 9848 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
 9849 \tabularnewline}%
 9850 }%

Backward compatible longraggedborder style.

9851 \compatglossarystyle{longraggedborder}{%
 9852 \csuse{@glscompstyle@longragged}%
 9853 }%

Backward compatible longraggedheader style.

```
9854 \compatglossarystyle{longraggedheader}{%  
9855 \csuse{@glscompstyle@longragged}%  
9856 }%
```

Backward compatible longraggedheaderborder style.

```
9857 \compatglossarystyle{longraggedheaderborder}{%  
9858 \csuse{@glscompstyle@longragged}%  
9859 }%
```

Backward compatible longragged3col style.

```
9860 \compatglossarystyle{longragged3col}{%  
9861 \renewcommand*{\glossaryentryfield}[5]{%  
9862 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%  
9863 \renewcommand*{\glossarysubentryfield}[6]{%  
9864 &  
9865 \glssubentryitem{##2}%  
9866 \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%  
9867 }%
```

Backward compatible longragged3colborder style.

```
9868 \compatglossarystyle{longragged3colborder}{%  
9869 \csuse{@glscompstyle@longragged3col}%  
9870 }%
```

Backward compatible longragged3colheader style.

```
9871 \compatglossarystyle{longragged3colheader}{%  
9872 \csuse{@glscompstyle@longragged3col}%  
9873 }%
```

Backward compatible longragged3colheaderborder style.

```
9874 \compatglossarystyle{longragged3colheaderborder}{%  
9875 \csuse{@glscompstyle@longragged3col}%  
9876 }%
```

Backward compatible altlongragged4col style.

```
9877 \compatglossarystyle{altlongragged4col}{%  
9878 \renewcommand*{\glossaryentryfield}[5]{%  
9879 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%  
9880 \renewcommand*{\glossarysubentryfield}[6]{%  
9881 &  
9882 \glssubentryitem{##2}%  
9883 \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%  
9884 }%
```

Backward compatible altlongragged4colheader style.

```
9885 \compatglossarystyle{altlongragged4colheader}{%  
9886 \csuse{@glscompstyle@altlong4col}%  
9887 }%
```

Backward compatible altlongragged4colborder style.

```
9888 \compatglossarystyle{altlongragged4colborder}{%
```

```

9889 \csuse{@glscompstyle@altlong4col}%
9890 }%

```

Backward compatible altlongragged4colheaderborder style.

```

9891 \compatglossarystyle{altlongragged4colheaderborder}{%
9892 \csuse{@glscompstyle@altlong4col}%
9893 }%

```

Backward compatible index style.

```

9894 \compatglossarystyle{index}{%
9895 \renewcommand*{\glossaryentryfield}[5]{%
9896 \item\glsentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9897 \ifx\relax##4\relax
9898 \else
9899 \space{##4}%
9900 \fi
9901 \space ##3\glspostdescription \space ##5}%
9902 \renewcommand*{\glossarysubentryfield}[6]{%
9903 \ifcase##1\relax
9904 % level 0
9905 \item
9906 \or
9907 % level 1
9908 \subitem
9909 \glssubentryitem{##2}%
9910 \else
9911 % all other levels
9912 \subsubitem
9913 \fi
9914 \textbf{\glstarget{##2}{##3}}%
9915 \ifx\relax##5\relax
9916 \else
9917 \space{##5}%
9918 \fi
9919 \space##4\glspostdescription\space ##6}%
9920 }%

```

Backward compatible indexgroup style.

```

9921 \compatglossarystyle{indexgroup}{%
9922 \csuse{@glscompstyle@index}%
9923 }%

```

Backward compatible indexhypergroup style.

```

9924 \compatglossarystyle{indexhypergroup}{%
9925 \csuse{@glscompstyle@index}%
9926 }%

```

Backward compatible tree style.

```

9927 \compatglossarystyle{tree}{%
9928 \renewcommand{\glossaryentryfield}[5]{%
9929 \hangindent0pt\relax

```

```

9930 \parindent0pt\relax
9931 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9932 \ifx\relax##4\relax
9933 \else
9934 \space{##4}%
9935 \fi
9936 \space ##3\glspostdescription \space ##5\par}%
9937 \renewcommand{\glossarysubentryfield}[6]{%
9938 \hangindent##1\glstreeindent\relax
9939 \parindent##1\glstreeindent\relax
9940 \ifnum##1=1\relax
9941 \glssubentryitem{##2}%
9942 \fi
9943 \textbf{\glstarget{##2}{##3}}%
9944 \ifx\relax##5\relax
9945 \else
9946 \space{##5}%
9947 \fi
9948 \space##4\glspostdescription\space ##6\par}%
9949 }%

```

Backward compatible treegroup style.

```

9950 \compatglossarystyle{treegroup}{%
9951 \csuse{@glscmpstyle@tree}%
9952 }%

```

Backward compatible treehypergroup style.

```

9953 \compatglossarystyle{treehypergroup}{%
9954 \csuse{@glscmpstyle@tree}%
9955 }%

```

Backward compatible treenoname style.

```

9956 \compatglossarystyle{treenoname}{%
9957 \renewcommand{\glossaryentryfield}[5]{%
9958 \hangindent0pt\relax
9959 \parindent0pt\relax
9960 \glstentryitem{##1}\textbf{\glstarget{##1}{##2}}%
9961 \ifx\relax##4\relax
9962 \else
9963 \space{##4}%
9964 \fi
9965 \space ##3\glspostdescription \space ##5\par}%
9966 \renewcommand{\glossarysubentryfield}[6]{%
9967 \hangindent##1\glstreeindent\relax
9968 \parindent##1\glstreeindent\relax
9969 \ifnum##1=1\relax
9970 \glssubentryitem{##2}%
9971 \fi
9972 \glstarget{##2}{\strut}%
9973 ##4\glspostdescription\space ##6\par}%
9974 }%

```

Backward compatible treenonamegroup style.

```
9975 \compatglossarystyle{treenonamegroup}{%
9976   \csuse{@glscompstyle@treenoname}%
9977 }%
```

Backward compatible treenonamehypergroup style.

```
9978 \compatglossarystyle{treenonamehypergroup}{%
9979   \csuse{@glscompstyle@treenoname}%
9980 }%
```

Backward compatible alttree style.

```
9981 \compatglossarystyle{alttree}{%
9982   \renewcommand{\glossaryentryfield}[5]{%
9983     \ifnum\@gls@prevlevel=0\relax
9984     \else
9985       \settowidth{\glstreeindent}{\textbf{\@glswidestname\space}}%
9986       \hangindent\glstreeindent
9987       \parindent\glstreeindent
9988     \fi
9989     \makebox[0pt][r]{\makebox[\glstreeindent][l]{%
9990       \glssentryitem{##1}\textbf{\glstarget{##1}{##2}}}%
9991     \ifx\relax##4\relax
9992     \else
9993       (##4)\space
9994     \fi
9995     ##3\glspostdescription \space ##5\par
9996     \def\@gls@prevlevel{0}%
9997   }%
9998   \renewcommand{\glossarysubentryfield}[6]{%
9999     \ifnum##1=1\relax
10000       \glssubentryitem{##2}%
10001     \fi
10002     \ifnum\@gls@prevlevel=##1\relax
10003     \else
10004       \@ifundefined{@glswidestname\romannumeral##1}{%
10005         \settowidth{\gls@tmplen}{\textbf{\@glswidestname\space}}{%
10006         \settowidth{\gls@tmplen}{\textbf{%
10007           \csname @glswidestname\romannumeral##1\endcsname\space}}}%
10008       \ifnum\@gls@prevlevel<##1\relax
10009         \setlength\glstreeindent\gls@tmplen
10010         \addtolength\glstreeindent\parindent
10011         \parindent\glstreeindent
10012       \else
10013         \@ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
10014           \settowidth{\glstreeindent}{\textbf{%
10015             \@glswidestname\space}}{%
10016           \settowidth{\glstreeindent}{\textbf{%
10017             \csname @glswidestname\romannumeral\@gls@prevlevel
10018             \endcsname\space}}}%
10019           \addtolength\parindent{-\glstreeindent}%

```



```

10020      \setlength\glstreeindent\parindent
10021      \fi
10022      \fi
10023      \hangindent\glstreeindent
10024      \makebox[0pt][r]{\makebox[\glstemplen][l]{%
10025        \textbf{\glstarget{##2}{##3}}}%
10026      \ifx##5\relax\relax
10027      \else
10028        (##5)\space
10029      \fi
10030      ##4\glspostdescription\space ##6\par
10031      \def\@gls@prevlevel{##1}%
10032    }%
10033 }%

```

Backward compatible alttreegroup style.

```

10034 \compatglossarystyle{alttreegroup}{%
10035   \csuse{@glscompstyle@almtree}%
10036 }%

```

Backward compatible almtreehypergroup style.

```

10037 \compatglossarystyle{almtreehypergroup}{%
10038   \csuse{@glscompstyle@almtree}%
10039 }%

```

Backward compatible mcolindex style.

```

10040 \compatglossarystyle{mcolindex}{%
10041   \csuse{@glscompstyle@index}%
10042 }%

```

Backward compatible mcolindexgroup style.

```

10043 \compatglossarystyle{mcolindexgroup}{%
10044   \csuse{@glscompstyle@index}%
10045 }%

```

Backward compatible mcolindexhypergroup style.

```

10046 \compatglossarystyle{mcolindexhypergroup}{%
10047   \csuse{@glscompstyle@index}%
10048 }%

```

Backward compatible mcoltree style.

```

10049 \compatglossarystyle{mcoltree}{%
10050   \csuse{@glscompstyle@tree}%
10051 }%

```

Backward compatible mcoltreegroup style.

```

10052 \compatglossarystyle{mcolindextreegroup}{%
10053   \csuse{@glscompstyle@tree}%
10054 }%

```

Backward compatible mcoltreehypergroup style.

```

10055 \compatglossarystyle{mcolindextreehypergroup}{%

```

```
10056 \csuse{@glscompstyle@tree}%
10057 }%
```

Backward compatible mcoltreenoname style.

```
10058 \compatglossarystyle{mcoltreenoname}{%
10059 \csuse{@glscompstyle@tree}%
10060 }%
```

Backward compatible mcoltreenonamegroup style.

```
10061 \compatglossarystyle{mcoltreenonamegroup}{%
10062 \csuse{@glscompstyle@tree}%
10063 }%
```

Backward compatible mcoltreenonamehypergroup style.

```
10064 \compatglossarystyle{mcoltreenonamehypergroup}{%
10065 \csuse{@glscompstyle@tree}%
10066 }%
```

Backward compatible mcolalmtree style.

```
10067 \compatglossarystyle{mcolalmtree}{%
10068 \csuse{@glscompstyle@almtree}%
10069 }%
```

Backward compatible mcolalmtreegroup style.

```
10070 \compatglossarystyle{mcolalmtreegroup}{%
10071 \csuse{@glscompstyle@almtree}%
10072 }%
```

Backward compatible mcolalmtreehypergroup style.

```
10073 \compatglossarystyle{mcolalmtreehypergroup}{%
10074 \csuse{@glscompstyle@almtree}%
10075 }%
```

Backward compatible superragged style.

```
10076 \compatglossarystyle{superragged}{%
10077 \renewcommand*{\glossaryentryfield}[5]{%
10078 \glsentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
10079 \tabularnewline}%
10080 \renewcommand*{\glossarysubentryfield}[6]{%
10081 &
10082 \glssubentryitem{##2}%
10083 \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
10084 \tabularnewline}%
10085 }%
```

Backward compatible superraggedborder style.

```
10086 \compatglossarystyle{superraggedborder}{%
10087 \csuse{@glscompstyle@superragged}%
10088 }%
```

Backward compatible superraggedheader style.

```
10089 \compatglossarystyle{superraggedheader}{%
10090 \csuse{@glscompstyle@superragged}%
10091 }%
```

Backward compatible superraggedheaderborder style.

```
10092 \compatglossarystyle{superraggedheaderborder}{%
10093   \csuse{@glscompstyle@superragged}%
10094 }%
```

Backward compatible superragged3col style.

```
10095 \compatglossarystyle{superragged3col}{%
10096   \renewcommand*{\glossaryentryfield}[5]{%
10097     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
10098   \renewcommand*{\glossarysubentryfield}[6]{%
10099     &
10100     \glssubentryitem{##2}%
10101     \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
10102 }%
```

Backward compatible superragged3colborder style.

```
10103 \compatglossarystyle{superragged3colborder}{%
10104   \csuse{@glscompstyle@superragged3col}%
10105 }%
```

Backward compatible superragged3colheader style.

```
10106 \compatglossarystyle{superragged3colheader}{%
10107   \csuse{@glscompstyle@superragged3col}%
10108 }%
```

Backward compatible superragged3colheaderborder style.

```
10109 \compatglossarystyle{superragged3colheaderborder}{%
10110   \csuse{@glscompstyle@superragged3col}%
10111 }%
```

Backward compatible altsuperragged4col style.

```
10112 \compatglossarystyle{altsuperragged4col}{%
10113   \renewcommand*{\glossaryentryfield}[5]{%
10114     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
10115   \renewcommand*{\glossarysubentryfield}[6]{%
10116     &
10117     \glssubentryitem{##2}%
10118     \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
10119 }%
```

Backward compatible altsuperragged4colheader style.

```
10120 \compatglossarystyle{altsuperragged4colheader}{%
10121   \csuse{@glscompstyle@altsuperragged4col}%
10122 }%
```

Backward compatible altsuperragged4colborder style.

```
10123 \compatglossarystyle{altsuperragged4colborder}{%
10124   \csuse{@glscompstyle@altsuperragged4col}%
10125 }%
```

Backward compatible altsuperragged4colheaderborder style.

```
10126 \compatglossarystyle{altsuperragged4colheaderborder}{%
```

```
10127 \csuse{@glscompstyle@altsuperragged4col}%
10128 }%
```

Backward compatible super style.

```
10129 \compatglossarystyle{super}{%
10130 \renewcommand*{\glossaryentryfield}[5]{%
10131 \glentryitem{##1}\glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
10132 \renewcommand*{\glossarysubentryfield}[6]{%
10133 &
10134 \glssubentryitem{##2}%
10135 \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
10136 }%
```

Backward compatible superborder style.

```
10137 \compatglossarystyle{superborder}{%
10138 \csuse{@glscompstyle@super}%
10139 }%
```

Backward compatible superheader style.

```
10140 \compatglossarystyle{superheader}{%
10141 \csuse{@glscompstyle@super}%
10142 }%
```

Backward compatible superheaderborder style.

```
10143 \compatglossarystyle{superheaderborder}{%
10144 \csuse{@glscompstyle@super}%
10145 }%
```

Backward compatible super3col style.

```
10146 \compatglossarystyle{super3col}{%
10147 \renewcommand*{\glossaryentryfield}[5]{%
10148 \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##5\\}%
10149 \renewcommand*{\glossarysubentryfield}[6]{%
10150 &
10151 \glssubentryitem{##2}%
10152 \glstarget{##2}{\strut}##4 & ##6\\}%
10153 }%
```

Backward compatible super3colborder style.

```
10154 \compatglossarystyle{super3colborder}{%
10155 \csuse{@glscompstyle@super3col}%
10156 }%
```

Backward compatible super3colheader style.

```
10157 \compatglossarystyle{super3colheader}{%
10158 \csuse{@glscompstyle@super3col}%
10159 }%
```

Backward compatible super3colheaderborder style.

```
10160 \compatglossarystyle{super3colheaderborder}{%
10161 \csuse{@glscompstyle@super3col}%
10162 }%
```

Backward compatible super4col style.

```
10163 \compatglossarystyle{super4col}{%
10164   \renewcommand*{\glossaryentryfield}[5]{%
10165     \glentryitem{##1}\glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
10166   \renewcommand*{\glossarysubentryfield}[6]{%
10167     &
10168     \glssubentryitem{##2}%
10169     \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
10170 }%
```

Backward compatible super4colheader style.

```
10171 \compatglossarystyle{super4colheader}{%
10172   \csuse{@glscompstyle@super4col}%
10173 }%
```

Backward compatible super4colborder style.

```
10174 \compatglossarystyle{super4colborder}{%
10175   \csuse{@glscompstyle@super4col}%
10176 }%
```

Backward compatible super4colheaderborder style.

```
10177 \compatglossarystyle{super4colheaderborder}{%
10178   \csuse{@glscompstyle@super4col}%
10179 }%
```

Backward compatible altsuper4col style.

```
10180 \compatglossarystyle{altsuper4col}{%
10181   \csuse{@glscompstyle@super4col}%
10182 }%
```

Backward compatible altsuper4colheader style.

```
10183 \compatglossarystyle{altsuper4colheader}{%
10184   \csuse{@glscompstyle@super4col}%
10185 }%
```

Backward compatible altsuper4colborder style.

```
10186 \compatglossarystyle{altsuper4colborder}{%
10187   \csuse{@glscompstyle@super4col}%
10188 }%
```

Backward compatible altsuper4colheaderborder style.

```
10189 \compatglossarystyle{altsuper4colheaderborder}{%
10190   \csuse{@glscompstyle@super4col}%
10191 }%
```

5 Accessibility Support (glossaries-accsupp Code)

The package is experimental. It is intended to provide a means of using the PDF accessibility support in glossary entries. See the documentation for further details about accessibility support.

```
10192 \NeedsTeXFormat{LaTeX2e}
```

Package version number now in line with main glossaries package number.

```
10193 \ProvidesPackage{glossaries-accsupp}[2017/01/19 v4.29 (NLCT)]
```

```
10194 Experimental glossaries accessibility]
```

Pass all options to glossaries:

```
10195 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
```

Process options:

```
10196 \ProcessOptions
```

This package should be loaded before glossaries-extra, so complain if that has already been loaded.

```
10197 \@ifpackageloaded{glossaries-extra}
```

```
10198 {%
```

If the accsupp option was used, \glsxtr@doaccsupp will have been set, otherwise it will be empty.

```
10199 \ifx\glsxtr@doaccsupp\empty
```

```
10200 \GlossariesWarning{The 'glossaries-accsupp'
```

```
10201 package has been loaded\MessageBreak
```

```
10202 after the 'glossaries-extra' package. This\MessageBreak
```

```
10203 can cause a failure to integrate both packages. \MessageBreak
```

```
10204 Either use the 'accsupp' option when you load\MessageBreak
```

```
10205 'glossaries-extra' or load 'glossaries-accsupp'\MessageBreak
```

```
10206 before loading 'glossaries-extra'}%
```

```
10207 \fi
```

```
10208 }
```

```
10209 {}
```

tibleglossentry Override style compatibility macros:

```
10210 \def\compatibleglossentry#1#2{%
```

```
10211 \toks@{#2}%
```

```
10212 \protected@edef\do@glossentry{%
```

```
10213 \noexpand\accsuppglossaryentryfield{#1}%
```

```
10214 {\noexpand\glsnamefont
```

```
10215 {\expandafter\expandonce\csname glo@\glsdetoklabel{#1}@name\endcsname}}%
```

```

10216    {\expandafter\expandonce\csname glo@glstetoklabel{#1}@desc\endcsname}%
10217    {\expandafter\expandonce\csname glo@glstetoklabel{#1}@symbol\endcsname}%
10218    {\the\toks@}%
10219    }%
10220    \@do@glossentry
10221 }

```

lesubglossentry

```

10222 \def\compatiblesubglossentry#1#2#3{%
10223   \toks@{#3}%
10224   \protected@edef\@do@subglossentry{%
10225     \noexpand\accsuppglossarysubentryfield{\number#1}%
10226     {#2}%
10227     {\noexpand\glsnamefont
10228      {\expandafter\expandonce\csname glo@glstetoklabel{#2}@name\endcsname}}}%
10229     {\expandafter\expandonce\csname glo@glstetoklabel{#2}@desc\endcsname}%
10230     {\expandafter\expandonce\csname glo@glstetoklabel{#2}@symbol\endcsname}%
10231     {\the\toks@}%
10232   }%
10233   \@do@subglossentry
10234 }

```

Required packages:

```

10235 \RequirePackage{glossaries}
10236 \RequirePackage{accsupp}

```

5.1 Defining Replacement Text

The version 0.1 stored the replacement text in the symbol key. This has been changed to use the new keys defined here. Example of use:

```
\newglossaryentry{dr}{name=Dr,description={},access={Doctor}}
```

access The replacement text corresponding to the name key:

```

10237 \define@key{glossentry}{access}{%
10238   \def\@glo@access{#1}%
10239 }

```

textaccess The replacement text corresponding to the text key:

```

10240 \define@key{glossentry}{textaccess}{%
10241   \def\@glo@textaccess{#1}%
10242 }

```

firstaccess The replacement text corresponding to the first key:

```

10243 \define@key{glossentry}{firstaccess}{%
10244   \def\@glo@firstaccess{#1}%
10245 }

```

pluralaccess The replacement text corresponding to the plural key:

```

10246 \define@key{glossentry}{pluralaccess}{%
10247   \def\@glo@pluralaccess{#1}%
10248 }
```

firstpluralaccess The replacement text corresponding to the firstplural key:

```

10249 \define@key{glossentry}{firstpluralaccess}{%
10250   \def\@glo@firstpluralaccess{#1}%
10251 }
```

symbolaccess The replacement text corresponding to the symbol key:

```

10252 \define@key{glossentry}{symbolaccess}{%
10253   \def\@glo@symbolaccess{#1}%
10254 }
```

symbolpluralaccess The replacement text corresponding to the symbolplural key:

```

10255 \define@key{glossentry}{symbolpluralaccess}{%
10256   \def\@glo@symbolpluralaccess{#1}%
10257 }
```

descriptionaccess The replacement text corresponding to the description key:

```

10258 \define@key{glossentry}{descriptionaccess}{%
10259   \def\@glo@descaccess{#1}%
10260 }
```

descriptionpluralaccess The replacement text corresponding to the descriptionplural key:

```

10261 \define@key{glossentry}{descriptionpluralaccess}{%
10262   \def\@glo@descpluralaccess{#1}%
10263 }
```

shortaccess The replacement text corresponding to the short key:

```

10264 \define@key{glossentry}{shortaccess}{%
10265   \def\@glo@shortaccess{#1}%
10266 }
```

shortpluralaccess The replacement text corresponding to the shortplural key:

```

10267 \define@key{glossentry}{shortpluralaccess}{%
10268   \def\@glo@shortpluralaccess{#1}%
10269 }
```

longaccess The replacement text corresponding to the long key:

```

10270 \define@key{glossentry}{longaccess}{%
10271   \def\@glo@longaccess{#1}%
10272 }
```

longpluralaccess The replacement text corresponding to the longplural key:

```

10273 \define@key{glossentry}{longpluralaccess}{%
10274   \def\@glo@longpluralaccess{#1}%
10275 }
```


There are no equivalent keys for the user1...user6 keys. The replacement text would have to be explicitly put in the value, e.g., user1={\glsaccsupp{inches}{in}}.

Append these new keys to \@gls@keymap:

```
10276 \appto\@gls@keymap{,%
10277   {access}{access},%
10278   {textaccess}{textaccess},%
10279   {firstaccess}{firstaccess},%
10280   {pluralaccess}{pluralaccess},%
10281   {firstpluralaccess}{firstpluralaccess},%
10282   {symbolaccess}{symbolaccess},%
10283   {symbolpluralaccess}{symbolpluralaccess},%
10284   {descaccess}{descaccess},%
10285   {descpluralaccess}{descpluralaccess},%
10286   {shortaccess}{shortaccess},%
10287   {shortpluralaccess}{shortpluralaccess},%
10288   {longaccess}{longaccess},%
10289   {longpluralaccess}{longpluralaccess}}%
10290 }
```

\@gls@noaccess Indicates that no replacement text has been provided.

```
10291 \def\@gls@noaccess{\relax}
```

Add to the start hook (the access key is initialised to the value of the symbol key at the start for backwards compatibility):

```
10292 \let\@gls@oldnewglossaryentryprehook\@newglossaryentryprehook
10293 \renewcommand*{\@newglossaryentryprehook}{%
10294   \@gls@oldnewglossaryentryprehook
10295   \def\@glo@access{\@glo@symbol}}%
```

Initialise the other keys:

```
10296 \def\@glo@textaccess{\@glo@access}%
10297 \def\@glo@firstaccess{\@glo@access}%
10298 \def\@glo@pluralaccess{\@glo@textaccess}%
10299 \def\@glo@firstpluralaccess{\@glo@pluralaccess}%
10300 \def\@glo@symbolaccess{\relax}%
10301 \def\@glo@symbolpluralaccess{\@glo@symbolaccess}%
10302 \def\@glo@descaccess{\relax}%
10303 \def\@glo@descpluralaccess{\@glo@descaccess}%
10304 \def\@glo@shortaccess{\relax}%
10305 \def\@glo@shortpluralaccess{\@glo@shortaccess}%
10306 \def\@glo@longaccess{\relax}%
10307 \def\@glo@longpluralaccess{\@glo@longaccess}%
10308 }
```

Add to the end hook:

```
10309 \let\@gls@oldnewglossaryentryposthook\@newglossaryentryposthook
10310 \renewcommand*{\@newglossaryentryposthook}{%
10311   \@gls@oldnewglossaryentryposthook}
```

Store the access information:

```

10312 \expandafter
10313 \protected@xdef\csname glo@\@glo@label @access\endcsname{%
10314 \@glo@access}%
10315 \expandafter
10316 \protected@xdef\csname glo@\@glo@label @textaccess\endcsname{%
10317 \@glo@textaccess}%
10318 \expandafter
10319 \protected@xdef\csname glo@\@glo@label @firstaccess\endcsname{%
10320 \@glo@firstaccess}%
10321 \expandafter
10322 \protected@xdef\csname glo@\@glo@label @pluralaccess\endcsname{%
10323 \@glo@pluralaccess}%
10324 \expandafter
10325 \protected@xdef\csname glo@\@glo@label @firstpluralaccess\endcsname{%
10326 \@glo@firstpluralaccess}%
10327 \expandafter
10328 \protected@xdef\csname glo@\@glo@label @symbolaccess\endcsname{%
10329 \@glo@symbolaccess}%
10330 \expandafter
10331 \protected@xdef\csname glo@\@glo@label @symbolpluralaccess\endcsname{%
10332 \@glo@symbolpluralaccess}%
10333 \expandafter
10334 \protected@xdef\csname glo@\@glo@label @descaccess\endcsname{%
10335 \@glo@descaccess}%
10336 \expandafter
10337 \protected@xdef\csname glo@\@glo@label @descpluralaccess\endcsname{%
10338 \@glo@descpluralaccess}%
10339 \expandafter
10340 \protected@xdef\csname glo@\@glo@label @shortaccess\endcsname{%
10341 \@glo@shortaccess}%
10342 \expandafter
10343 \protected@xdef\csname glo@\@glo@label @shortpluralaccess\endcsname{%
10344 \@glo@shortpluralaccess}%
10345 \expandafter
10346 \protected@xdef\csname glo@\@glo@label @longaccess\endcsname{%
10347 \@glo@longaccess}%
10348 \expandafter
10349 \protected@xdef\csname glo@\@glo@label @longpluralaccess\endcsname{%
10350 \@glo@longpluralaccess}%
10351 }

```

5.2 Accessing Replacement Text

`\glsentryaccess` Get the value of the access key for the entry with the given label:

```

10352 \newcommand*{\glsentryaccess}[1]{%
10353 \@gls@entry@field{#1}{access}%
10354 }

```

entrytextaccess Get the value of the textaccess key for the entry with the given label:

```
10355 \newcommand*{\glentrytextaccess}[1]{%
10356   \@gls@entry@field{#1}{textaccess}%
10357 }
```

entryfirstaccess Get the value of the firstaccess key for the entry with the given label:

```
10358 \newcommand*{\glentryfirstaccess}[1]{%
10359   \@gls@entry@field{#1}{firstaccess}%
10360 }
```

entrypluralaccess Get the value of the pluralaccess key for the entry with the given label:

```
10361 \newcommand*{\glentrypluralaccess}[1]{%
10362   \@gls@entry@field{#1}{pluralaccess}%
10363 }
```

entryfirstpluralaccess Get the value of the firstpluralaccess key for the entry with the given label:

```
10364 \newcommand*{\glentryfirstpluralaccess}[1]{%
10365   \csname glo@#1@firstpluralaccess\endcsname
10366 }
```

entrysymbolaccess Get the value of the symbolaccess key for the entry with the given label:

```
10367 \newcommand*{\glentrysymbolaccess}[1]{%
10368   \@gls@entry@field{#1}{symbolaccess}%
10369 }
```

entrysymbolpluralaccess Get the value of the symbolpluralaccess key for the entry with the given label:

```
10370 \newcommand*{\glentrysymbolpluralaccess}[1]{%
10371   \@gls@entry@field{#1}{symbolpluralaccess}%
10372 }
```

entrydescaccess Get the value of the descriptionaccess key for the entry with the given label:

```
10373 \newcommand*{\glentrydescaccess}[1]{%
10374   \@gls@entry@field{#1}{descaccess}%
10375 }
```

entrydescpluralaccess Get the value of the descriptionpluralaccess key for the entry with the given label:

```
10376 \newcommand*{\glentrydescpluralaccess}[1]{%
10377   \@gls@entry@field{#1}{descaccess}%
10378 }
```

entryshortaccess Get the value of the shortaccess key for the entry with the given label:

```
10379 \newcommand*{\glentryshortaccess}[1]{%
10380   \@gls@entry@field{#1}{shortaccess}%
10381 }
```

entryshortpluralaccess Get the value of the shortpluralaccess key for the entry with the given label:

```
10382 \newcommand*{\glentryshortpluralaccess}[1]{%
10383   \@gls@entry@field{#1}{shortpluralaccess}%
10384 }
```

entrylongaccess Get the value of the longaccess key for the entry with the given label:

```
10385 \newcommand*{\glsentrylongaccess}[1]{%
10386   \@gls@entry@field{#1}{longaccess}%
10387 }
```

ongpluralaccess Get the value of the longpluralaccess key for the entry with the given label:

```
10388 \newcommand*{\glsentrylongpluralaccess}[1]{%
10389   \@gls@entry@field{#1}{longpluralaccess}%
10390 }
```

\glsaccsupp \glsaccsupp{<replacement text>}{<text>}

This can be redefined to use E or Alt instead of ActualText. (I don't have the software to test the E or Alt options.)

```
10391 \newcommand*{\glsaccsupp}[2]{%
10392   \BeginAccSupp{ActualText=#1}#2\EndAccSupp{}%
10393 }
```

\xglsaccsupp Fully expands replacement text before calling \glsaccsupp

```
10394 \newcommand*{\xglsaccsupp}[2]{%
10395   \protected@edef\@gls@replacementtext{#1}%
10396   \expandafter\glsaccsupp\expandafter{\@gls@replacementtext}{#2}%
10397 }
```

@access@display

```
10398 \newcommand*{\@gls@access@display}[2]{%
10399   \protected@edef\@glo@access{#2}%
10400   \ifx\@glo@access\@gls@noaccess
10401     #1%
10402   \else
10403     \xglsaccsupp{\@glo@access}{#1}%
10404   \fi
10405 }
```

meaccessdisplay Displays the first argument with the accessibility text for the entry with the label given by the second argument (if set).

```
10406 \DeclareRobustCommand*{\glsnameaccessdisplay}[2]{%
10407   \@gls@access@display{#1}{\glsentryaccess{#2}}%
10408 }
```

xtaccessdisplay As above but for the textaccess replacement text.

```
10409 \DeclareRobustCommand*{\glsstextaccessdisplay}[2]{%
10410   \@gls@access@display{#1}{\glsentrytextaccess{#2}}%
10411 }
```

alaccessdisplay As above but for the pluralaccess replacement text.

```
10412 \DeclareRobustCommand*{\glspluralaccessdisplay}[2]{%
10413   \@gls@access@display{#1}{\glsentrypluralaccess{#2}}%
10414 }
```

staccessdisplay As above but for the firstaccess replacement text.

```
10415 \DeclareRobustCommand*\glfirstaccessdisplay}[2]{%
10416 \@gls@access@display{#1}{\glentryfirstaccess{#2}}%
10417 }
```

alaccessdisplay As above but for the firstpluralaccess replacement text.

```
10418 \DeclareRobustCommand*\glfirstpluralaccessdisplay}[2]{%
10419 \@gls@access@display{#1}{\glentryfirstpluralaccess{#2}}%
10420 }
```

olaccessdisplay As above but for the symbolaccess replacement text.

```
10421 \DeclareRobustCommand*\glssymbolaccessdisplay}[2]{%
10422 \@gls@access@display{#1}{\glentrysymbolaccess{#2}}%
10423 }
```

alaccessdisplay As above but for the symbolpluralaccess replacement text.

```
10424 \DeclareRobustCommand*\glssymbolpluralaccessdisplay}[2]{%
10425 \@gls@access@display{#1}{\glentrysymbolpluralaccess{#2}}%
10426 }
```

onaccessdisplay As above but for the descriptionaccess replacement text.

```
10427 \DeclareRobustCommand*\glsdescriptionaccessdisplay}[2]{%
10428 \@gls@access@display{#1}{\glentrydescaccess{#2}}%
10429 }
```

alaccessdisplay As above but for the descriptionpluralaccess replacement text.

```
10430 \DeclareRobustCommand*\glsdescriptionpluralaccessdisplay}[2]{%
10431 \@gls@access@display{#1}{\glentrydescpluraccess{#2}}%
10432 }
```

rtaccessdisplay As above but for the shortaccess replacement text.

```
10433 \DeclareRobustCommand*\glsshortaccessdisplay}[2]{%
10434 \@gls@access@display{#1}{\glentryshortaccess{#2}}%
10435 }
```

alaccessdisplay As above but for the shortpluralaccess replacement text.

```
10436 \DeclareRobustCommand*\glsshortpluralaccessdisplay}[2]{%
10437 \@gls@access@display{#1}{\glentryshortpluralaccess{#2}}%
10438 }
```

ngaccessdisplay As above but for the longaccess replacement text.

```
10439 \DeclareRobustCommand*\glslongaccessdisplay}[2]{%
10440 \@gls@access@display{#1}{\glentrylongaccess{#2}}%
10441 }
```

alaccessdisplay As above but for the longpluralaccess replacement text.

```
10442 \DeclareRobustCommand*\glslongpluralaccessdisplay}[2]{%
10443 \@gls@access@display{#1}{\glentrylongpluralaccess{#2}}%
10444 }
```

`\saccessdisplay` Gets the replacement text corresponding to the named key given by the first argument and calls the appropriate command defined above.

```

10445 \DeclareRobustCommand*\saccessdisplay}[3]{%
10446   \@ifundefined{gls#1accessdisplay}%
10447   {%
10448     \PackageError{glossaries-accsupp}{No accessibility support
10449       for key ‘#1’}{}%
10450   }%
10451   {%
10452     \csname gls#1accessdisplay\endcsname{#2}{#3}%
10453   }%
10454 }

```

`\default@entryfmt` Redefine the default entry format to use accessibility information

```

10455 \renewcommand*\@gls@default@entryfmt}[2]{%
10456   \ifdefempty\glscustomtext
10457   {%
10458     \glsifplural
10459     {%

```

Plural form

```

10460     \glscapscase
10461     {%

```

Don't adjust case

```

10462     \ifglsused\glslabel
10463     {%

```

Subsequent use

```

10464         #2{\glspluralaccessdisplay
10465           {\glsentryplural{\glslabel}}{\glslabel}}%
10466         {\glsdescriptionpluralaccessdisplay
10467           {\glsentrydescplural{\glslabel}}{\glslabel}}%
10468         {\glsymbolpluralaccessdisplay
10469           {\glsentrysymbolplural{\glslabel}}{\glslabel}}
10470         {\glsinsert}%
10471     }%
10472     {%

```

First use

```

10473         #1{\glsfirstpluralaccessdisplay
10474           {\glsentryfirstplural{\glslabel}}{\glslabel}}%
10475         {\glsdescriptionpluralaccessdisplay
10476           {\glsentrydescplural{\glslabel}}{\glslabel}}%
10477         {\glsymbolpluralaccessdisplay
10478           {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10479         {\glsinsert}%
10480     }%
10481     {%
10482     {%

```

Make first letter upper case

```
10483      \ifglsused\glslabel
10484      {%
```

Subsequent use.

```
10485      #2{\glspluralaccessdisplay
10486          {\Glsentryplural{\glslabel}}{\glslabel}}%
10487          {\glsdescriptionpluralaccessdisplay
10488              {\glsentrydescplural{\glslabel}}{\glslabel}}%
10489          {\glssymbolpluralaccessdisplay
10490              {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10491          {\glsinsert}}%
10492      }%
10493      {%
```

First use

```
10494      #1{\glsfirstpluralaccessdisplay
10495          {\Glsentryfirstplural{\glslabel}}{\glslabel}}%
10496          {\glsdescriptionpluralaccessdisplay
10497              {\glsentrydescplural{\glslabel}}{\glslabel}}%
10498          {\glssymbolpluralaccessdisplay
10499              {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10500          {\glsinsert}}%
10501      }%
10502      }%
10503      {%
```

Make all upper case

```
10504      \ifglsused\glslabel
10505      {%
```

Subsequent use

```
10506      \MakeUppercase{%
10507      #2{\glspluralaccessdisplay
10508          {\glsentryplural{\glslabel}}{\glslabel}}%
10509          {\glsdescriptionpluralaccessdisplay
10510              {\glsentrydescplural{\glslabel}}{\glslabel}}%
10511          {\glssymbolpluralaccessdisplay
10512              {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
10513          {\glsinsert}}%
10514      }%
10515      {%
```

First use

```
10516      \MakeUppercase{%
10517      #1{\glsfirstpluralaccessdisplay
10518          {\glsentryfirstplural{\glslabel}}{\glslabel}}%
10519          {\glsdescriptionpluralaccessdisplay
10520              {\glsentrydescplural{\glslabel}}{\glslabel}}%
10521          {\glssymbolpluralaccessdisplay
10522              {\glsentrysymbolplural{\glslabel}}{\glslabel}}%
```

```

10523         {\glsinsert}}}%
10524     }%
10525 }%
10526 }%
10527 {%

```

Singular form

```

10528     \glscapscase
10529     {%

```

Don't adjust case

```

10530     \ifglsused\glslabel
10531     {%

```

Subsequent use

```

10532     #2{\glstextaccessdisplay
10533         {\glsentrytext{\glslabel}}{\glslabel}}%
10534     {\glsdescriptionaccessdisplay
10535         {\glsentrydesc{\glslabel}}{\glslabel}}%
10536     {\glssymbolaccessdisplay
10537         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10538     {\glsinsert}}%
10539 }%
10540 {%

```

First use

```

10541     #1{\glsfirstaccessdisplay
10542         {\glsentryfirst{\glslabel}}{\glslabel}}%
10543     {\glsdescriptionaccessdisplay
10544         {\glsentrydesc{\glslabel}}{\glslabel}}%
10545     {\glssymbolaccessdisplay
10546         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10547     {\glsinsert}}%
10548 }%
10549 }%
10550 {%

```

Make first letter upper case

```

10551     \ifglsused\glslabel
10552     {%

```

Subsequent use

```

10553     #2{\glstextaccessdisplay
10554         {\Glsentrytext{\glslabel}}{\glslabel}}%
10555     {\glsdescriptionaccessdisplay
10556         {\glsentrydesc{\glslabel}}{\glslabel}}%
10557     {\glssymbolaccessdisplay
10558         {\glsentrysymbol{\glslabel}}{\glslabel}}%
10559     {\glsinsert}}%
10560 }%
10561 {%

```


First use

```

10562      #1{\glsfirstaccessdisplay
10563          {\Glsentryfirst{\glslabel}}{\glslabel}}%
10564          {\glsdescriptionaccessdisplay
10565              {\glsentrydesc{\glslabel}}{\glslabel}}%
10566          {\glsymbolaccessdisplay
10567              {\glsentrysymbol{\glslabel}}{\glslabel}}%
10568          {\glsinsert}}%
10569      }%
10570  }%
10571  {%

```

Make all upper case

```

10572      \ifglsused\glslabel
10573      {%

```

Subsequent use

```

10574      \MakeUppercase{%
10575          #2{\glstextaccessdisplay
10576              {\glsentrytext{\glslabel}}{\glslabel}}%
10577              {\glsdescriptionaccessdisplay
10578                  {\glsentrydesc{\glslabel}}{\glslabel}}%
10579                  {\glsymbolaccessdisplay
10580                      {\glsentrysymbol{\glslabel}}{\glslabel}}%
10581                      {\glsinsert}}}%
10582      }%
10583      {%

```

First use

```

10584      \MakeUppercase{%
10585          #1{\glsfirstaccessdisplay
10586              {\glsentryfirst{\glslabel}}{\glslabel}}%
10587              {\glsdescriptionaccessdisplay
10588                  {\glsentrydesc{\glslabel}}{\glslabel}}%
10589                  {\glsymbolaccessdisplay
10590                      {\glsentrysymbol{\glslabel}}{\glslabel}}%
10591                      {\glsinsert}}}%
10592      }%
10593  }%
10594  }%
10595  }%
10596  {%

```

Custom text provided in \glsdisp

```

10597      \ifglsused{\glslabel}%
10598      {%

```

Subsequent use

```

10599      #2{\glscustomtext}%
10600      {\glsdescriptionaccessdisplay
10601          {\glsentrydesc{\glslabel}}{\glslabel}}%

```

```

10602      {\glssymbolaccessdisplay
10603      {\glentrysymbol{\glslabel}}{\glslabel}}%
10604      {\glsinsert}%
10605  }%
10606  {%

```

First use

```

10607      #1{\glscustomtext}%
10608      {\glsdescriptionaccessdisplay
10609      {\glentrydesc{\glslabel}}{\glslabel}}%
10610      {\glssymbolaccessdisplay
10611      {\glentrysymbol{\glslabel}}{\glslabel}}%
10612      {\glsinsert}%
10613  }%
10614  }%
10615 }

```

`\glsgenentryfmt` Redefine to use accessibility information.

```

10616 \renewcommand*{\glsgenentryfmt}{%
10617   \ifdefempty\glscustomtext
10618   {%
10619     \glsifplural
10620     {%

```

Plural form

```

10621     \glscapscase
10622     {%

```

Don't adjust case

```

10623     \ifglused\glslabel
10624     {%

```

Subsequent use

```

10625     \glspluralaccessdisplay
10626     {\glentryplural{\glslabel}}{\glslabel}%
10627     \glsinsert
10628   }%
10629   {%

```

First use

```

10630     \glsfirstpluralaccessdisplay
10631     {\glentryfirstplural{\glslabel}}{\glslabel}%
10632     \glsinsert
10633   }%
10634   }%
10635   {%

```

Make first letter upper case

```

10636     \ifglused\glslabel
10637     {%

```

Subsequent use.

```
10638      \glspluralaccessdisplay
10639      {\Glsentryplural{\glslabel}}{\glslabel}%
10640      \glsinsert
10641      }%
10642      {%
```

First use

```
10643      \glsfirstpluralaccessdisplay
10644      {\Glsentryfirstplural{\glslabel}}{\glslabel}%
10645      \glsinsert
10646      }%
10647      }%
10648      {%
```

Make all upper case

```
10649      \ifglused\glslabel
10650      {%
```

Subsequent use

```
10651      \glspluralaccessdisplay
10652      {\mfirstucMakeUppercase{\Glsentryplural{\glslabel}}}%
10653      {\glslabel}%
10654      \mfirstucMakeUppercase{\glsinsert}%
10655      }%
10656      {%
```

First use

```
10657      \glsfirstpluralaccessdisplay
10658      {\mfirstucMakeUppercase{\Glsentryfirstplural{\glslabel}}}%
10659      {\glslabel}%
10660      \mfirstucMakeUppercase{\glsinsert}%
10661      }%
10662      }%
10663      }%
10664      {%
```

Singular form

```
10665      \glscapscase
10666      {%
```

Don't adjust case

```
10667      \ifglused\glslabel
10668      {%
```

Subsequent use

```
10669      \glstextaccessdisplay{\Glsentrytext{\glslabel}}{\glslabel}%
10670      \glsinsert
10671      }%
10672      {%
```

First use

```

10673      \glsfirstaccessdisplay{\glsentryfirst{\glslabel}}{\glslabel}%
10674      \glsinsert
10675      }%
10676      }%
10677      {%

```

Make first letter upper case

```

10678      \ifglsused\glslabel
10679      {%

```

Subsequent use

```

10680      \glstextaccessdisplay{\Glsentrytext{\glslabel}}{\glslabel}%
10681      \glsinsert
10682      }%
10683      {%

```

First use

```

10684      \glsfirstaccessdisplay{\Glsentryfirst{\glslabel}}{\glslabel}%
10685      \glsinsert
10686      }%
10687      }%
10688      {%

```

Make all upper case

```

10689      \ifglsused\glslabel
10690      {%

```

Subsequent use

```

10691      \glstextaccessdisplay
10692      {\mfirstucMakeUppercase{\glsentrytext{\glslabel}}}{\glslabel}%
10693      \mfirstucMakeUppercase{\glsinsert}%
10694      }%
10695      {%

```

First use

```

10696      \glsfirstaccessdisplay
10697      {\mfirstucMakeUppercase{\glsentryfirst{\glslabel}}}{\glslabel}%
10698      \mfirstucMakeUppercase{\glsinsert}%
10699      }%
10700      }%
10701      }%
10702      }%
10703      {%

```

Custom text provided in `\glsdisp`. (The insert should be empty at this point.) The accessibility information, if required, will have to be explicitly included in the custom text.

```

10704      \glscustomtext\glsinsert
10705      }%
10706      }

```

`\glsgenacfmt` Redefine to include accessibility information.

```
10707 \renewcommand*{\glsgenacfmt}{%
10708   \ifdefempty\glscustomtext
10709     {%
10710       \ifglused\glslabel
10711       {%
```

Subsequent use:

```
10712     \glsifplural
10713     {%
```

Subsequent plural form:

```
10714     \glscapscase
10715     {%
```

Subsequent plural form, don't adjust case:

```
10716     \acronymfont
10717     {\glsshortpluralaccessdisplay
10718      {\glentryshortpl{\glslabel}}{\glslabel}}%
10719     \glsinsert
10720   }%
10721   {%
```

Subsequent plural form, make first letter upper case:

```
10722     \acronymfont
10723     {\glsshortpluralaccessdisplay
10724      {\Glsentryshortpl{\glslabel}}{\glslabel}}%
10725     \glsinsert
10726   }%
10727   {%
```

Subsequent plural form, all caps:

```
10728     \mfirstucMakeUppercase
10729     {\acronymfont
10730      {\glsshortpluralaccessdisplay
10731       {\glentryshortpl{\glslabel}}{\glslabel}}%
10732      \glsinsert}%
10733   }%
10734   }%
10735   {%
```

Subsequent singular form

```
10736     \glscapscase
10737     {%
```

Subsequent singular form, don't adjust case:

```
10738     \acronymfont
10739     {\glsshortaccessdisplay{\glentryshort{\glslabel}}{\glslabel}}%
10740     \glsinsert
10741   }%
10742   {%
```

Subsequent singular form, make first letter upper case:

```
10743      \acronymfont
10744      {\glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
10745      \glsinsert
10746      }%
10747      {%
```

Subsequent singular form, all caps:

```
10748      \mfirstucMakeUppercase
10749      {\acronymfont{%
10750      \glsshortaccessdisplay{\Glsentryshort{\glslabel}}{\glslabel}}%
10751      \glsinsert}%
10752      }%
10753      }%
10754      }%
10755      {%
```

First use:

```
10756      \glsifplural
10757      {%
```

First use plural form:

```
10758      \glscapscase
10759      {%
```

First use plural form, don't adjust case:

```
10760      \genplacrfullformat{\glslabel}{\glsinsert}%
10761      }%
10762      {%
```

First use plural form, make first letter upper case:

```
10763      \Genplacrfullformat{\glslabel}{\glsinsert}%
10764      }%
10765      {%
```

First use plural form, all caps:

```
10766      \mfirstucMakeUppercase
10767      {\genplacrfullformat{\glslabel}{\glsinsert}}%
10768      }%
10769      }%
10770      {%
```

First use singular form

```
10771      \glscapscase
10772      {%
```

First use singular form, don't adjust case:

```
10773      \genacrfullformat{\glslabel}{\glsinsert}%
10774      }%
10775      {%
```

First use singular form, make first letter upper case:

```
10776      \Genacrfullformat{\glslabel}{\glsinsert}%
10777      }%
10778      {%
```

First use singular form, all caps:

```
10779      \mfirstucMakeUppercase
10780      {\genacrfullformat{\glslabel}{\glsinsert}}%
10781      }%
10782      }%
10783      }%
10784      }%
10785      {%
```

User supplied text. (The insert should be empty at this point.) The accessibility information, if required, will have to be explicitly included in the custom text.

```
10786      \glscustomtext
10787      }%
10788 }
```

enacrfullformat Redefine to include accessibility information.

```
10789 \renewcommand*{\genacrfullformat}[2]{%
10790   \glslongaccessdisplay{\glsentrylong{#1}}{#1}#2\space
10791   (\glsshortaccessdisplay{\protect\firstacronymfont{\glsentryshort{#1}}}{#1}}%
10792 }
```

enacrfullformat Redefine to include accessibility information.

```
10793 \renewcommand*{\Genacrfullformat}[2]{%
10794   \glslongaccessdisplay{\Glsentrylong{#1}}{#1}#2\space
10795   (\glsshortaccessdisplay{\protect\firstacronymfont{\Glsentryshort{#1}}}{#1}}%
10796 }
```

placrfullformat Redefine to include accessibility information.

```
10797 \renewcommand*{\genplacrfullformat}[2]{%
10798   \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}#2\space
10799   (\glsshortpluralaccessdisplay
10800    {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1}}%
10801 }
```

placrfullformat Redefine to include accessibility information.

```
10802 \renewcommand*{\Genplacrfullformat}[2]{%
10803   \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}#2\space
10804   (\glsshortpluralaccessdisplay
10805    {\protect\firstacronymfont{\glsentryshortpl{#1}}}{#1}}%
10806 }
```

\@acrshort

```
10807 \def\@acrshort#1#2[#3]{%
10808   \glsdoifexists{#2}%
```

```

10809 {%
10810   \let\do@gl@s@link@checkfirsthyper\relax

10811   \let\gl@sifplural\@secondoftwo
10812   \let\gl@scapscase\@firstofthree
10813   \let\gl@sinsert\@empty
10814   \def\glscustomtext{%
10815     \acronymfont{\glsshortaccessdisplay{\glentryshort{#2}}{#2}}#3%
10816   }%

   Call \@gl@s@link
10817   \@gl@s@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10818 }%

10819 \glspostlinkhook
10820 }

```

\@Acrshort

```

10821 \def\@Acrshort#1#2[#3]{%
10822   \glstoifexists{#2}%
10823   {%
10824     \let\do@gl@s@link@checkfirsthyper\relax

10825     \let\gl@sifplural\@secondoftwo
10826     \let\gl@scapscase\@secondofthree
10827     \let\gl@sinsert\@empty
10828     \def\glscustomtext{%
10829       \acronymfont{\glsshortaccessdisplay{\Glsentryshort{#2}}{#2}}#3%
10830     }%

     Call \@gl@s@link
10831     \@gl@s@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10832   }%

10833   \glspostlinkhook
10834 }

```

\@ACRshort

```

10835 \def\@ACRshort#1#2[#3]{%
10836   \glstoifexists{#2}%
10837   {%
10838     \let\do@gl@s@link@checkfirsthyper\relax

10839     \let\gl@sifplural\@secondoftwo
10840     \let\gl@scapscase\@thirdofthree
10841     \let\gl@sinsert\@empty
10842     \def\glscustomtext{%
10843       \acronymfont{\glsshortaccessdisplay
10844         {\MakeUppercase{\glentryshort{#2}}}{#2}}#3%
10845     }%

```



```

Call \@gls@link
10846   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10847   }%

10848   \glspostlinkhook
10849 }

```

\@acrlong

```

10850 \def\@acrlong#1#2[#3]{%
10851   \glsdoifexists{#2}%
10852   {%
10853     \let\do@gls@link@checkfirsthyper\relax

10854     \let\glsifplural\@secondoftwo
10855     \let\glscapscase\@firstofthree
10856     \let\glsinsert\@empty
10857     \def\glscustomtext{%
10858       \acronymfont{\glslongaccessdisplay{\glsentrylong{#2}}{#2}}#3%
10859     }%

```

```

Call \@gls@link
10860   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10861   }%

10862   \glspostlinkhook
10863 }

```

\@Acrlong

```

10864 \def\@Acrlong#1#2[#3]{%
10865   \glsdoifexists{#2}%
10866   {%
10867     \let\do@gls@link@checkfirsthyper\relax

10868     \let\glsifplural\@secondoftwo
10869     \let\glscapscase\@firstofthree
10870     \let\glsinsert\@empty
10871     \def\glscustomtext{%
10872       \acronymfont{\glslongaccessdisplay{\Glsentrylong{#2}}{#2}}#3%
10873     }%

```

```

Call \@gls@link
10874   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10875   }%

10876   \glspostlinkhook
10877 }

```

\@ACRlong

```

10878 \def\@ACRlong#1#2[#3]{%
10879   \glsdoifexists{#2}%
10880   {%
10881     \let\do@gls@link@checkfirsthyper\relax

```

```

10882 \let\glsifplural\@secondoftwo
10883 \let\glsifcaps\@firstofthree
10884 \let\glsinsert\@empty
10885 \def\glscustomtext{%
10886   \acronymfont{\glslongaccessdisplay{%
10887     \MakeUppercase{\glsentrylong{#2}}}{#2}#3}%
10888   }%

  Call \@gls@link
10889   \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
10890 }%

10891 \glspostlinkhook
10892 }

```

5.3 Displaying the Glossary

We need to redefine the way the glossary entries are formatted to include the accessibility support. The predefined glossary styles use `\glossentryname`, `\glossentrydesc` and `\glossentrysymbol`, but we need to provide compatibility with earlier versions in case users have defined their own styles using `\accsuppglossaryentryfield` and `\accsuppglossarysubentryfield`.

Now redefine `\glossentryname`, `\glossentrydesc` and `\glossentrysymbol` etc so they use the accessibility stuff.

```

10893 \renewcommand*{\glossentryname}[1]{%
10894   \glsdoifexists{#1}%
10895   {%
10896     \glsnamefont{\glsnameaccessdisplay{\glsentryname{#1}}{#1}}%
10897   }%
10898 }

10899 \renewcommand*{\glossentrydesc}[1]{%
10900   \glsdoifexists{#1}%
10901   {%
10902     \glsnamefont{\glsnameaccessdisplay{\Glsentryname{#1}}{#1}}%
10903   }%
10904 }

10905 \renewcommand*{\glossentrydesc}[1]{%
10906   \glsdoifexists{#1}%
10907   {%
10908     \glsdescriptionaccessdisplay{\glsentrydesc{#1}}{#1}%
10909   }%
10910 }

10911 \renewcommand*{\Glossentrydesc}[1]{%
10912   \glsdoifexists{#1}%
10913   {%
10914     \glsdescriptionaccessdisplay{\Glsentrydesc{#1}}{#1}%
10915   }%
10916 }

```

```

10917 \renewcommand*{\glossentrysymbol}[1]{%
10918   \glsdoifexists{#1}%
10919   {%
10920     \glssymbolaccessdisplay{\glsentrysymbol{#1}}{#1}%
10921   }%
10922 }

10923 \renewcommand*{\Glossentrysymbol}[1]{%
10924   \glsdoifexists{#1}%
10925   {%
10926     \glssymbolaccessdisplay{\Glsentrysymbol{#1}}{#1}%
10927   }%
10928 }

```

ssaryentryfield

```

10929 \newcommand*{\accsuppglossaryentryfield}[5]{%
10930   \glossaryentryfield{#1}%
10931   {\glsnameaccessdisplay{#2}{#1}}%
10932   {\glsdescriptionaccessdisplay{#3}{#1}}%
10933   {\glssymbolaccessdisplay{#4}{#1}}{#5}%
10934 }

```

rysubentryfield

```

10935 \newcommand*{\accsuppglossarysubentryfield}[6]{%
10936   \glossarysubentryfield{#1}{#2}%
10937   {\glsnameaccessdisplay{#3}{#2}}%
10938   {\glsdescriptionaccessdisplay{#4}{#2}}%
10939   {\glssymbolaccessdisplay{#5}{#2}}{#6}%
10940 }

```

5.4 Acronyms

Redefine acronym styles provided by glossaries:

long-short *<long>* (*<short>*) acronym style.

```

10941 \renewacronymstyle{long-short}%
10942 {%

```

Check for long form in case this is a mixed glossary.

```

10943   \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
10944 }%
10945 {%
10946   \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10947   \renewcommand*{\genacrfullformat}[2]{%
10948     \glslongaccessdisplay{\glsentrylong{##1}}{##1}##2\space
10949     (\glsshortaccessdisplay
10950       {\protect\firstacronymfont{\glsentryshort{##1}}}{##1})%
10951   }%
10952   \renewcommand*{\Genacrfullformat}[2]{%

```

```

10953 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}##2\space
10954 (\glsshortaccessdisplay
10955   {\protect\firstacronymfont{\glsentryshort{##1}}{##1}})%
10956 }%
10957 \renewcommand*{\genplacrfullformat}[2]{%
10958   \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}##2\space
10959   (\glsshortpluralaccessdisplay
10960     {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}})%
10961   }%
10962 \renewcommand*{\Genplacrfullformat}[2]{%
10963   \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}##2\space
10964   (\glsshortpluralaccessdisplay
10965     {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}})%
10966   }%
10967 \renewcommand*{\acronymentry}[1]{%
10968   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}
10969 \renewcommand*{\acronymsort}[2]{##1}%
10970 \renewcommand*{\acronymfont}[1]{##1}%
10971 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
10972 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
10973 }

```

short-long (*short*) (*long*) acronym style.

```

10974 \renewacronymstyle{short-long}%
10975 {%

```

Check for long form in case this is a mixed glossary.

```

10976 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
10977 }%
10978 {%
10979 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
10980 \renewcommand*{\genacrfullformat}[2]{%
10981   \glsshortaccessdisplay
10982     {\protect\firstacronymfont{\glsentryshort{##1}}{##1}##2\space
10983     (\glslongaccessdisplay{\glsentrylong{##1}}{##1}})%
10984   }%
10985 \renewcommand*{\Genacrfullformat}[2]{%
10986   \glsshortaccessdisplay
10987     {\protect\firstacronymfont{\Glsentryshort{##1}}{##1}##2\space
10988     (\glslongaccessdisplay{\glsentrylong{##1}}{##1}})%
10989   }%
10990 \renewcommand*{\genplacrfullformat}[2]{%
10991   \glsshortpluralaccessdisplay
10992     {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}##2\space
10993     (\glslongpluralaccessdisplay
10994       {\glsentrylongpl{##1}}{##1}})%
10995   }%
10996 \renewcommand*{\Genplacrfullformat}[2]{%
10997   \glsshortpluralaccessdisplay
10998     {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2\space

```

```

10999 (\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1})%
11000 }%
11001 \renewcommand*{\acronymentry}[1]{%
11002   \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
11003 \renewcommand*{\acronymsort}[2]{##1}%
11004 \renewcommand*{\acronymfont}[1]{##1}%
11005 \renewcommand*{\firstacronymfont}[1]{\acronymfont{##1}}%
11006 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11007 }

```

long-short-desc *<long>* (*<short>*) acronym style that has an accompanying description (which the user needs to supply).

```

11008 \renewacronymstyle{long-short-desc}%
11009 {%
11010   \GlsUseAcrEntryDisplayStyle{long-short}%
11011 }%
11012 {%
11013   \GlsUseAcrStyleDefs{long-short}%
11014   \renewcommand*{\GenericAcronymFields}{}%
11015   \renewcommand*{\acronymsort}[2]{##2}%
11016   \renewcommand*{\acronymentry}[1]{%
11017     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11018     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11019 }

```

g-sc-short-desc *<long>* (\textsc{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

11020 \renewacronymstyle{long-sc-short-desc}%
11021 {%
11022   \GlsUseAcrEntryDisplayStyle{long-sc-short}%
11023 }%
11024 {%
11025   \GlsUseAcrStyleDefs{long-sc-short}%
11026   \renewcommand*{\GenericAcronymFields}{}%
11027   \renewcommand*{\acronymsort}[2]{##2}%
11028   \renewcommand*{\acronymentry}[1]{%
11029     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11030     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11031 }

```

g-sm-short-desc *<long>* (\textsmaller{<short>}) acronym style that has an accompanying description (which the user needs to supply).

```

11032 \renewacronymstyle{long-sm-short-desc}%
11033 {%
11034   \GlsUseAcrEntryDisplayStyle{long-sm-short}%
11035 }%
11036 {%
11037   \GlsUseAcrStyleDefs{long-sm-short}%
11038   \renewcommand*{\GenericAcronymFields}{}%

```

```

11039 \renewcommand*{\acronymsort}[2]{##2}%
11040 \renewcommand*{\acronymentry}[1]{%
11041   \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11042   (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11043 }

```

short-long-desc *<short>* (*<long>*) acronym style that has an accompanying description (which the user needs to supply).

```

11044 \renewacronymstyle{short-long-desc}%
11045 {%
11046   \GlsUseAcrEntryDispStyle{short-long}%
11047 }%
11048 {%
11049   \GlsUseAcrStyleDefs{short-long}%
11050   \renewcommand*{\GenericAcronymFields}{}%
11051   \renewcommand*{\acronymsort}[2]{##2}%
11052   \renewcommand*{\acronymentry}[1]{%
11053     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11054     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11055 }

```

short-long-desc *<long>* (*\textsc{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

11056 \renewacronymstyle{sc-short-long-desc}%
11057 {%
11058   \GlsUseAcrEntryDispStyle{sc-short-long}%
11059 }%
11060 {%
11061   \GlsUseAcrStyleDefs{sc-short-long}%
11062   \renewcommand*{\GenericAcronymFields}{}%
11063   \renewcommand*{\acronymsort}[2]{##2}%
11064   \renewcommand*{\acronymentry}[1]{%
11065     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11066     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11067 }

```

short-long-desc *<long>* (*\textsmaller{<short>}*) acronym style that has an accompanying description (which the user needs to supply).

```

11068 \renewacronymstyle{sm-short-long-desc}%
11069 {%
11070   \GlsUseAcrEntryDispStyle{sm-short-long}%
11071 }%
11072 {%
11073   \GlsUseAcrStyleDefs{sm-short-long}%
11074   \renewcommand*{\GenericAcronymFields}{}%
11075   \renewcommand*{\acronymsort}[2]{##2}%
11076   \renewcommand*{\acronymentry}[1]{%
11077     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11078     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%

```

11079 }

dua *<long>* only acronym style.

11080 \renewacronymstyle{dua}%
11081 {%

Check for long form in case this is a mixed glossary.

11082 \ifdefempty\glscustomtext
11083 {%
11084 \ifglshaslong{\glslabel}%
11085 {%
11086 \glsifplural
11087 {%

Plural form:

11088 \glscapscase
11089 {%

Plural form, don't adjust case:

11090 \gslongpluralaccessdisplay{\glentrylongpl{\glslabel}}{\glslabel}%
11091 \glsinsert
11092 }%
11093 {%

Plural form, make first letter upper case:

11094 \gslongpluralaccessdisplay{\Glentrylongpl{\glslabel}}{\glslabel}%
11095 \glsinsert
11096 }%
11097 {%

Plural form, all caps:

11098 \gslongpluralaccessdisplay
11099 {\mfirstucMakeUppercase{\glentrylongpl{\glslabel}}}{\glslabel}%
11100 \mfirstucMakeUppercase{\glsinsert}%
11101 }%
11102 }%
11103 {%

Singular form

11104 \glscapscase
11105 {%

Singular form, don't adjust case:

11106 \gslongaccessdisplay{\glentrylong{\glslabel}}{\glslabel}\glsinsert
11107 }%
11108 {%

Subsequent singular form, make first letter upper case:

11109 \gslongaccessdisplay{\Glentrylong{\glslabel}}{\glslabel}\glsinsert
11110 }%
11111 {%

Subsequent singular form, all caps:

```

11112      \glslongaccessdisplay
11113      {\mfirstucMakeUppercase
11114        {\glsentrylong{\glslabel}\glsinsert}}{\glslabel}%
11115      \mfirstucMakeUppercase{\glsinsert}%
11116    }%
11117  }%
11118 }%
11119 {%

```

Not an acronym:

```

11120      \glsgenentryfmt
11121    }%
11122 }%
11123 {\glscustomtext\glsinsert}%
11124}%
11125{%
11126 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%
11127 \renewcommand*{\acrfullfmt}[3]{%
11128   \glslink[##1]{##2}{%
11129     \glslongaccessdisplay{\glsentrylong{##2}}{##2}##3\space
11130     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
11131 \renewcommand*{\Acrfullfmt}[3]{%
11132   \glslink[##1]{##2}{%
11133     \glslongaccessdisplay{\Glsentrylong{##2}}{##2}##3\space
11134     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
11135 \renewcommand*{\ACRfullfmt}[3]{%
11136   \glslink[##1]{##2}{%
11137     \glslongaccessdisplay
11138     {\mfirstucMakeUppercase{\glsentrylong{##2}}{##2}##3\space
11139     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}}{##2}})%
11140 \renewcommand*{\acrfullplfmt}[3]{%
11141   \glslink[##1]{##2}{%
11142     \glslongpluralaccessdisplay
11143     {\glsentrylongpl{##2}}{##2}##3\space
11144     (\glsshortpluralaccessdisplay
11145     {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
11146 \renewcommand*{\Acrfullplfmt}[3]{%
11147   \glslink[##1]{##2}{%
11148     \glslongpluralaccessdisplay
11149     {\Glsentrylongpl{##2}}{##2}##3\space
11150     (\glsshortpluralaccessdisplay
11151     {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
11152 \renewcommand*{\ACRfullplfmt}[3]{%
11153   \glslink[##1]{##2}{%
11154     \glslongpluralaccessdisplay
11155     {\mfirstucMakeUppercase{\glsentrylongpl{##2}}{##2}##3\space
11156     (\glsshortpluralaccessdisplay
11157     {\acronymfont{\glsentryshortpl{##2}}}{##2}})%
11158 \renewcommand*{\glsentryfull}[1]{%

```



```

11159 \glslongaccessdisplay{\glsentrylong{##1}}\space
11160 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
11161 }%
11162 \renewcommand*{\Glsentryfull}[1]{%
11163 \glslongaccessdisplay{\Glsentrylong{##1}}{##1}\space
11164 (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})%
11165 }%
11166 \renewcommand*{\glsentryfullpl}[1]{%
11167 \glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}\space
11168 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
11169 }%
11170 \renewcommand*{\Glsentryfullpl}[1]{%
11171 \glslongpluralaccessdisplay{\Glsentrylongpl{##1}}{##1}\space
11172 (\glsshortpluralaccessdisplay{\acronymfont{\glsentryshortpl{##1}}}{##1})%
11173 }%
11174 \renewcommand*{\acronymentry}[1]{%
11175 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1}}%
11176 \renewcommand*{\acronymsort}[2]{##1}%
11177 \renewcommand*{\acronymfont}[1]{##1}%
11178 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11179 }

```

dua-desc *<long>* only acronym style with user-supplied description.

```

11180 \renewacronymstyle{dua-desc}%
11181 {%
11182 \GlsUseAcrEntryDispStyle{dua}%
11183 }%
11184 {%
11185 \GlsUseAcrStyleDefs{dua}%
11186 \renewcommand*{\GenericAcronymFields}{}%
11187 \renewcommand*{\acronymentry}[1]{%
11188 \glslongaccessdisplay{\acronymfont{\glsentrylong{##1}}}{##1}}%
11189 \renewcommand*{\acronymsort}[2]{##2}%
11190 }%

```

footnote *<short>*\footnote{*<long>*} acronym style.

```

11191 \renewacronymstyle{footnote}%
11192 {%
11193 \ifglshaslong{\glslabel}{\glsgenacfmt}{\glsgenentryfmt}%
11194 }%
11195 {%
11196 \renewcommand*{\GenericAcronymFields}{description={\the\glslongtok}}%

```

Need to ensure hyperlinks are switched off on first use:

```

11197 \glshyperfirstfalse
11198 \renewcommand*{\genacrfullformat}[2]{%
11199 \glsshortaccessdisplay
11200 {\protect\firstacronymfont{\glsentryshort{##1}}}{##1}##2%

```

```

11201 \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
11202 }%
11203 \renewcommand*{\Genacrfullformat}[2]{%
11204 \glsshortaccessdisplay
11205   {\firstacronymfont{\Glsentryshort{##1}}{##1}##2%
11206 \protect\footnote{\glslongaccessdisplay{\glsentrylong{##1}}{##1}}%
11207 }%
11208 \renewcommand*{\genplacrfullformat}[2]{%
11209 \glsshortpluralaccessdisplay
11210   {\protect\firstacronymfont{\glsentryshortpl{##1}}{##1}##2%
11211 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
11212 }%
11213 \renewcommand*{\Genplacrfullformat}[2]{%
11214 \glsshortpluralaccessdisplay
11215   {\protect\firstacronymfont{\Glsentryshortpl{##1}}{##1}##2%
11216 \protect\footnote{\glslongpluralaccessdisplay{\glsentrylongpl{##1}}{##1}}%
11217 }%
11218 \renewcommand*{\acronymentry}[1]{%
11219 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}{##1}}%
11220 \renewcommand*{\acronymsort}[2]{##1}%
11221 \renewcommand*{\acronymfont}[1]{##1}%
11222 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%

```

Don't use footnotes for \acrfull:

```

11223 \renewcommand*{\acrfullfmt}[3]{%
11224 \glslink{##1}{##2}{%
11225 \glsshortaccessdisplay{\acronymfont{\glsentryshort{##2}}{##2}##3\space
11226 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
11227 \renewcommand*{\Acrfullfmt}[3]{%
11228 \glslink{##1}{##2}{%
11229 \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##2}}{##2}##3\space
11230 (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
11231 \renewcommand*{\ACRfullfmt}[3]{%
11232 \glslink{##1}{##2}{%
11233 \glsshortaccessdisplay
11234   {\mfirstucMakeUppercase
11235   {\acronymfont{\glsentryshort{##2}}{##2}##3\space
11236   (\glslongaccessdisplay{\glsentrylong{##2}}{##2})}}}%
11237 \renewcommand*{\acrfullplfmt}[3]{%
11238 \glslink{##1}{##2}{%
11239 \glsshortpluralaccessdisplay
11240   {\acronymfont{\glsentryshortpl{##2}}{##2}##3\space
11241   (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
11242 \renewcommand*{\Acrfullplfmt}[3]{%
11243 \glslink{##1}{##2}{%
11244 \glsshortpluralaccessdisplay
11245   {\acronymfont{\Glsentryshortpl{##2}}{##2}##3\space
11246   (\glslongpluralaccessdisplay{\glsentrylongpl{##2}}{##2})}}}%
11247 \renewcommand*{\ACRfullplfmt}[3]{%
11248 \glslink{##1}{##2}{%

```

```

11249 \glsshortpluralaccessdisplay
11250 {\mfirstucMakeUppercase
11251 {\acronymfont{\glentryshortpl{##2}}}{##2}##3\space
11252 (\glslongpluralaccessdisplay{\glentrylongpl{##2}}{##2}}}%

```

Similarly for \glentryfull etc:

```

11253 \renewcommand*{\glentryfull}[1]{%
11254 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}}{##1}\space
11255 (\glslongaccessdisplay{\glentrylong{##1}}{##1}}}%
11256 \renewcommand*{\Glsentryfull}[1]{%
11257 \glsshortaccessdisplay{\acronymfont{\Glsentryshort{##1}}}{##1}\space
11258 (\glslongaccessdisplay{\glentrylong{##1}}{##1}}}%
11259 \renewcommand*{\glentryfullpl}[1]{%
11260 \glsshortpluralaccessdisplay
11261 {\acronymfont{\glentryshortpl{##1}}}{##1}\space
11262 (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}}}%
11263 \renewcommand*{\Glsentryfullpl}[1]{%
11264 \glsshortpluralaccessdisplay
11265 {\acronymfont{\Glsentryshortpl{##1}}}{##1}\space
11266 (\glslongpluralaccessdisplay{\glentrylongpl{##1}}{##1}}}%
11267 }

```

footnote-sc \textsc{<short>}\footnote{<long>} acronym style.

```

11268 \renewacronymstyle{footnote-sc}%
11269 {%
11270 \GlsUseAcrEntryDispStyle{footnote}%
11271 }%
11272 {%
11273 \GlsUseAcrStyleDefs{footnote}%
11274 \renewcommand{\acronymentry}[1]{%
11275 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}}{##1}}
11276 \renewcommand{\acronymfont}[1]{\textsc{##1}}%
11277 \renewcommand*{\acrpluralsuffix}{\glstextup{\glspluralsuffix}}%
11278 }%

```

footnote-sm \textsmaller{<short>}\footnote{<long>} acronym style.

```

11279 \renewacronymstyle{footnote-sm}%
11280 {%
11281 \GlsUseAcrEntryDispStyle{footnote}%
11282 }%
11283 {%
11284 \GlsUseAcrStyleDefs{footnote}%
11285 \renewcommand{\acronymentry}[1]{%
11286 \glsshortaccessdisplay{\acronymfont{\glentryshort{##1}}}{##1}}
11287 \renewcommand{\acronymfont}[1]{\textsmaller{##1}}%
11288 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
11289 }%

```

footnote-desc <short>\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11290 \renewacronymstyle{footnote-desc}%
11291 {%
11292   \GlsUseAcrEntryDisplayStyle{footnote}%
11293 }%
11294 {%
11295   \GlsUseAcrStyleDefs{footnote}%
11296   \renewcommand*{\GenericAcronymFields}{}%
11297   \renewcommand*{\acronymsort}[2]{##2}%
11298   \renewcommand*{\acronymentry}[1]{%
11299     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11300     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11301 }

```

ootnote-sc-desc \textsc{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11302 \renewacronymstyle{footnote-sc-desc}%
11303 {%
11304   \GlsUseAcrEntryDisplayStyle{footnote-sc}%
11305 }%
11306 {%
11307   \GlsUseAcrStyleDefs{footnote-sc}%
11308   \renewcommand*{\GenericAcronymFields}{}%
11309   \renewcommand*{\acronymsort}[2]{##2}%
11310   \renewcommand*{\acronymentry}[1]{%
11311     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11312     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11313 }

```

ootnote-sm-desc \textsmaller{<short>}\footnote{<long>} acronym style that has an accompanying description (which the user needs to supply).

```

11314 \renewacronymstyle{footnote-sm-desc}%
11315 {%
11316   \GlsUseAcrEntryDisplayStyle{footnote-sm}%
11317 }%
11318 {%
11319   \GlsUseAcrStyleDefs{footnote-sm}%
11320   \renewcommand*{\GenericAcronymFields}{}%
11321   \renewcommand*{\acronymsort}[2]{##2}%
11322   \renewcommand*{\acronymentry}[1]{%
11323     \glslongaccessdisplay{\glsentrylong{##1}}{##1}\space
11324     (\glsshortaccessdisplay{\acronymfont{\glsentryshort{##1}}}{##1})}%
11325 }

```

Use \newacronymhook to modify the key list to set the access text to the long version by default.

```

11326 \renewcommand*{\newacronymhook}{%
11327   \edef\@gls@keylist{shortaccess=\the\glslongtok,%
11328     \the\glskeylisttok}%
11329   \expandafter\glskeylisttok\expandafter{\@gls@keylist}%

```

11330 }

1tNewAcronymDef Modify default style to use access text:

```
11331 \renewcommand*{\DefaultNewAcronymDef}{%
11332   \edef\@do@newglossaryentry{%
11333     \noexpand\newglossaryentry{\the\glslabeltok}%
11334     {%
11335       type=\acronymtype,%
11336       name={\the\glsshorttok},%
11337       description={\the\glslongtok},%
11338       descriptionaccess=\relax,
11339       text={\the\glsshorttok},%
11340       access={\noexpand\@glo@textaccess},%
11341       sort={\the\glsshorttok},%
11342       short={\the\glsshorttok},%
11343       shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11344       shortaccess={\the\glslongtok},%
11345       long={\the\glslongtok},%
11346       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11347       descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11348       first={\noexpand\glslongaccessdisplay
11349         {\the\glslongtok}{\the\glslabeltok}\space
11350         (\noexpand\glsshortaccessdisplay
11351           {\the\glsshorttok}{\the\glslabeltok})},%
11352       plural={\the\glsshorttok\acrpluralsuffix},%
11353       firstplural={\noexpand\glslongpluralaccessdisplay
11354         {\noexpand\@glo@longpl}{\the\glslabeltok}\space
11355         (\noexpand\glsshortpluralaccessdisplay
11356           {\noexpand\@glo@shortpl}{\the\glslabeltok})},%
11357       firstaccess=\relax,
11358       firstpluralaccess=\relax,
11359       textaccess={\noexpand\@glo@shortaccess},%
11360       \the\glskeylisttok
11361     }%
11362   }%
11363   \let\@org@gls@assign@firstpl\gls@assign@firstpl
11364   \let\@org@gls@assign@plural\gls@assign@plural
11365   \let\@org@gls@assign@descplural\gls@assign@descplural
11366   \def\gls@assign@firstpl##1##2{%
11367     \@gls@expand@field{##1}{firstpl}{##2}%
11368   }%
11369   \def\gls@assign@plural##1##2{%
11370     \@gls@expand@field{##1}{plural}{##2}%
11371   }%
11372   \def\gls@assign@descplural##1##2{%
11373     \@gls@expand@field{##1}{descplural}{##2}%
11374   }%
11375   \@do@newglossaryentry
11376   \let\gls@assign@firstpl\@org@gls@assign@firstpl
```

```

11377 \let\gls@assign@plural\@org@gls@assign@plural
11378 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11379 }

```

teNewAcronymDef

```

11380 \renewcommand*{\DescriptionFootnoteNewAcronymDef}{%
11381 \edef\@do@newglossaryentry{%
11382 \noexpand\newglossaryentry{\the\glslabeltok}%
11383 {%
11384 type=\acronymtype,%
11385 name={\noexpand\acronymfont{\the\glsshorttok}},%
11386 sort={\the\glsshorttok},%
11387 text={\the\glsshorttok},%
11388 short={\the\glsshorttok},%
11389 shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11390 shortaccess={\the\glslongtok},%
11391 long={\the\glslongtok},%
11392 longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11393 access={\noexpand\@glo@textaccess},%
11394 plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11395 symbol={\the\glslongtok},%
11396 symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11397 firstpluralaccess=\relax,
11398 textaccess={\noexpand\@glo@shortaccess},%
11399 \the\glskeylisttok
11400 }%
11401 }%
11402 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11403 \let\@org@gls@assign@plural\gls@assign@plural
11404 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11405 \def\gls@assign@firstpl##1##2{%
11406 \@@gls@expand@field{##1}{firstpl}{##2}%
11407 }%
11408 \def\gls@assign@plural##1##2{%
11409 \@@gls@expand@field{##1}{plural}{##2}%
11410 }%
11411 \def\gls@assign@symbolplural##1##2{%
11412 \@@gls@expand@field{##1}{symbolplural}{##2}%
11413 }%
11414 \do@newglossaryentry
11415 \let\gls@assign@plural\@org@gls@assign@plural
11416 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11417 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11418 }

```

onNewAcronymDef

```

11419 \renewcommand*{\DescriptionNewAcronymDef}{%
11420 \edef\@do@newglossaryentry{%
11421 \noexpand\newglossaryentry{\the\glslabeltok}%

```

```

11422 {%
11423     type=\acronymtype,%
11424     name={\noexpand
11425         \acrnameformat{\the\glsshorttok}{\the\glslongtok}},%
11426     access={\noexpand\@glo@textaccess},%
11427     sort={\the\glsshorttok},%
11428     short={\the\glsshorttok},%
11429     shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11430     shortaccess={\the\glslongtok},%
11431     long={\the\glslongtok},%
11432     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11433     first={\the\glslongtok},%
11434     firstaccess=\relax,
11435     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11436     text={\the\glsshorttok},%
11437     textaccess={\the\glslongtok},%
11438     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11439     symbol={\noexpand\@glo@text},%
11440     symbolaccess={\noexpand\@glo@textaccess},%
11441     symbolplural={\noexpand\@glo@plural},%
11442     firstpluralaccess=\relax,
11443     textaccess={\noexpand\@glo@shortaccess},%
11444     \the\glskeylisttok}%
11445 }%
11446 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11447 \let\@org@gls@assign@plural\gls@assign@plural
11448 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11449 \def\gls@assign@firstpl##1##2{%
11450     \@gls@expand@field{##1}{firstpl}{##2}%
11451 }%
11452 \def\gls@assign@plural##1##2{%
11453     \@gls@expand@field{##1}{plural}{##2}%
11454 }%
11455 \def\gls@assign@symbolplural##1##2{%
11456     \@gls@expand@field{##1}{symbolplural}{##2}%
11457 }%
11458 \do@newglossaryentry
11459 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11460 \let\gls@assign@plural\@org@gls@assign@plural
11461 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11462 }

```

teNewAcronymDef

```

11463 \renewcommand*{\FootnoteNewAcronymDef}{%
11464     \edef\@do@newglossaryentry{%
11465         \noexpand\newglossaryentry{\the\glslabeltok}%
11466         {%
11467             type=\acronymtype,%
11468             name={\noexpand\acronymfont{\the\glsshorttok}},%

```

```

11469     sort={\the\glssshorttok},%
11470     text={\the\glssshorttok},%
11471     textaccess={\the\glslongtok},%
11472     access={\noexpand\@glo@textaccess},%
11473     plural={\the\glssshorttok\noexpand\acrpluralsuffix},%
11474     short={\the\glssshorttok},%
11475     shortplural={\the\glssshorttok\noexpand\acrpluralsuffix},%
11476     long={\the\glslongtok},%
11477     longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11478     description={\the\glslongtok},%
11479     descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11480     \the\glskeylisttok
11481   }%
11482 }%
11483 \let\@org@gls@assign@plural\gls@assign@plural
11484 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11485 \let\@org@gls@assign@descplural\gls@assign@descplural
11486 \def\gls@assign@firstpl##1##2{%
11487   \@@gls@expand@field{##1}{firstpl}{##2}%
11488 }%
11489 \def\gls@assign@plural##1##2{%
11490   \@@gls@expand@field{##1}{plural}{##2}%
11491 }%
11492 \def\gls@assign@descplural##1##2{%
11493   \@@gls@expand@field{##1}{descplural}{##2}%
11494 }%
11495 \do@newglossaryentry
11496 \let\gls@assign@plural\@org@gls@assign@plural
11497 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11498 \let\gls@assign@descplural\@org@gls@assign@descplural
11499 }

```

11NewAcronymDef

```

11500 \renewcommand*{\SmallNewAcronymDef}{%
11501   \edef\@do@newglossaryentry{%
11502     \noexpand\newglossaryentry{\the\glslabeltok}%
11503     {%
11504       type=\acronymtype,%
11505       name={\noexpand\acronymfont{\the\glssshorttok}},%
11506       access={\noexpand\@glo@symbolaccess},%
11507       sort={\the\glssshorttok},%
11508       short={\the\glssshorttok},%
11509       shortplural={\the\glssshorttok\noexpand\acrpluralsuffix},%
11510       shortaccess={\the\glslongtok},%
11511       long={\the\glslongtok},%
11512       longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11513       text={\noexpand\@glo@short},%
11514       textaccess={\noexpand\@glo@shortaccess},%
11515       plural={\noexpand\@glo@shortpl},%

```



```

11516     first={\the\glslongtok},%
11517     firstaccess=\relax,
11518     firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
11519     description={\noexpand\@glo@first},%
11520     descriptionplural={\noexpand\@glo@firstplural},%
11521     symbol={\the\glsshorttok},%
11522     symbolaccess={\the\glslongtok},%
11523     symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
11524     \the\glskeylisttok
11525 }%
11526 }%
11527 \let\@org@gls@assign@firstpl\gls@assign@firstpl
11528 \let\@org@gls@assign@plural\gls@assign@plural
11529 \let\@org@gls@assign@descplural\gls@assign@descplural
11530 \let\@org@gls@assign@symbolplural\gls@assign@symbolplural
11531 \def\gls@assign@firstpl##1##2{%
11532   \@@gls@expand@field{##1}{firstpl}{##2}%
11533 }%
11534 \def\gls@assign@plural##1##2{%
11535   \@@gls@expand@field{##1}{plural}{##2}%
11536 }%
11537 \def\gls@assign@descplural##1##2{%
11538   \@@gls@expand@field{##1}{descplural}{##2}%
11539 }%
11540 \def\gls@assign@symbolplural##1##2{%
11541   \@@gls@expand@field{##1}{symbolplural}{##2}%
11542 }%
11543 \do@newglossaryentry
11544 \let\gls@assign@firstpl\@org@gls@assign@firstpl
11545 \let\gls@assign@plural\@org@gls@assign@plural
11546 \let\gls@assign@descplural\@org@gls@assign@descplural
11547 \let\gls@assign@symbolplural\@org@gls@assign@symbolplural
11548 }

```

The following are kept for compatibility with versions before 3.0:

sshortaccesskey

```
11549 \newcommand*{\glsshortaccesskey}{\glsshortkey access}%
```

pluralaccesskey

```
11550 \newcommand*{\glsshortpluralaccesskey}{\glsshortpluralkey access}%
```

lslongaccesskey

```
11551 \newcommand*{\glslongaccesskey}{\glslongkey access}%
```

pluralaccesskey

```
11552 \newcommand*{\glslongpluralaccesskey}{\glslongpluralkey access}%
```

5.5 Debugging Commands

owglonameaccess

```
11553 \newcommand*{\showglonameaccess}[1]{%
11554   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
11555 }
```

owglotextaccess

```
11556 \newcommand*{\showglotextaccess}[1]{%
11557   \expandafter\show\csname glo@\glsdetoklabel{#1}@textaccess\endcsname
11558 }
```

glopluralaccess

```
11559 \newcommand*{\showglopluralaccess}[1]{%
11560   \expandafter\show\csname glo@\glsdetoklabel{#1}@pluralaccess\endcsname
11561 }
```

wglofirstaccess

```
11562 \newcommand*{\showglofirstaccess}[1]{%
11563   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstaccess\endcsname
11564 }
```

rstpluralaccess

```
11565 \newcommand*{\showglofirstpluralaccess}[1]{%
11566   \expandafter\show\csname glo@\glsdetoklabel{#1}@firstpluralaccess\endcsname
11567 }
```

glosymbolaccess

```
11568 \newcommand*{\showglosymbolaccess}[1]{%
11569   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolaccess\endcsname
11570 }
```

bolpluralaccess

```
11571 \newcommand*{\showglosymbolpluralaccess}[1]{%
11572   \expandafter\show\csname glo@\glsdetoklabel{#1}@symbolpluralaccess\endcsname
11573 }
```

owglodescaccess

```
11574 \newcommand*{\showglodescaccess}[1]{%
11575   \expandafter\show\csname glo@\glsdetoklabel{#1}@descaccess\endcsname
11576 }
```

escpluralaccess

```
11577 \newcommand*{\showglodescpluralaccess}[1]{%
11578   \expandafter\show\csname glo@\glsdetoklabel{#1}@descpluralaccess\endcsname
11579 }
```

wgloshortaccess

```
11580 \newcommand*{\showgloshortaccess}[1]{%  
11581   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortaccess\endcsname  
11582 }
```

ortpluralaccess

```
11583 \newcommand*{\showgloshortpluralaccess}[1]{%  
11584   \expandafter\show\csname glo@\glsdetoklabel{#1}@shortpluralaccess\endcsname  
11585 }
```

owglolongaccess

```
11586 \newcommand*{\showglolongaccess}[1]{%  
11587   \expandafter\show\csname glo@\glsdetoklabel{#1}@longaccess\endcsname  
11588 }
```

ongpluralaccess

```
11589 \newcommand*{\showglolongpluralaccess}[1]{%  
11590   \expandafter\show\csname glo@\glsdetoklabel{#1}@longpluralaccess\endcsname  
11591 }
```

6 Multi-Lingual Support

Many thanks to everyone who contributed to the translations both via email and on comp.text.tex. Language support has now been split off into independent language modules.

```
11592 \NeedsTeXFormat{LaTeX2e}
11593 \ProvidesPackage{glossaries-babel}[2017/01/19 v4.29 (NLCT)]
```

Load tracklang to obtain language settings.

```
11594 \RequirePackage{tracklang}
11595 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11596 \AnyTrackedLanguages
11597 {%
11598   \ForEachTrackedDialect{\this@dialect}{%
11599     \IfTrackedLanguageFileExists{\this@dialect}%
11600     {glossaries-}% prefix
11601     {.ldf}%
11602     {%
11603       \RequireGlossariesLang{\CurrentTrackedTag}%
11604     }%
11605     {%
11606       \PackageWarningNoLine{glossaries}%
11607       {No language module detected for ‘\this@dialect’.\MessageBreak
11608        Language modules need to be installed separately.\MessageBreak
11609        Please check on CTAN for a bundle called\MessageBreak
11610        ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11611     }%
11612   }%
11613 }%
11614 {}%
```

6.1 Polyglossia Captions

Language support has now been split off into independent language modules.

```
11615 \NeedsTeXFormat{LaTeX2e}
11616 \ProvidesPackage{glossaries-polyglossia}[2017/01/19 v4.29 (NLCT)]
```

Load tracklang to obtain language settings.

```
11617 \RequirePackage{tracklang}
11618 \let\glsifusetranslator\@secondoftwo
```

Check for tracked languages:

```
11619 \AnyTrackedLanguages
```

```

11620 {%
11621     \ForEachTrackedDialect{\this@dialect}{%
11622         \IfTrackedLanguageFileExists{\this@dialect}%
11623         {glossaries-}% prefix
11624         {.ldf}%
11625         {%
11626             \RequireGlossariesLang{\CurrentTrackedTag}%
11627         }%
11628         {%
11629             \PackageWarningNoLine{glossaries}%
11630             {No language module detected for ‘\this@dialect’.\MessageBreak
11631             Language modules need to be installed separately.\MessageBreak
11632             Please check on CTAN for a bundle called\MessageBreak
11633             ‘glossaries-\CurrentTrackedLanguage’ or similar}%
11634         }%
11635     }%
11636 }%
11637 {}%

```

Glossary

`makeindex` An indexing application. [10](#), [25](#), [26](#), [173](#)

`xindy` An flexible indexing application with multilingual support written in Perl. [10](#), [25](#), [26](#), [173](#)

Change History

1.01 (2007-05-17)	numberline: numberline option added . . . 6
General: Added range facility in format key 109	1.12 (2008-03-08)
\writeist: Added spaces after \delimN and \delimR in ist file 155	\@GLSpl: now uses \glentrydescplural and \glentrysymbolplural instead of \glentrydesc and \glentrysymbol 123
1.04 (2007-08-03)	\@Glspl@: now uses \glentrydescplural and \glentrysymbolplural instead of \glentrydesc and \glentrysymbol 122
1.05 (2007-08-10)	\@glsp1@: now uses \glentrydescplural and \glentrysymbolplural instead of \glentrydesc and \glentrysymbol 121
\glossarysection: added \@mkboth to \glossarysection 37	General: added check for \hypertarget separate to \hyperlink (memoir defines \hyperlink but not \hypertarget) 117
\gls@defglossaryentry: Changed the default value of the sort key to just the value of the name key 78	descriptionplural: new 60
1.07 (2007-09-13)	\gls@defglossaryentry: Changed default first plural to be first key with s appended (was text key with s appended) 77
\@gls@link: fixed bug caused by \theglentrycounter setting the page number too soon 107	descriptionplural support added 77
\glsadd: fixed bug caused by \theglentrycounter setting the page number too soon 153	symbolplural support added 77
1.08 (2007-10-13)	\Glsentrydescplural: New 146
General: Added babel support 31	\Glsentrydescplural: New 146
listgroup: changed listgroup style to use \glsgetgrouptitle 267	\Glsentrysymbolplural: New 147
altlistgroup: changed altlistgroup style to use \glsgetgrouptitle 268	\Glsentrysymbolplural: New 147
1.1 (2008-02-22)	\SetDescriptionFootnoteAcronymStyle: Added \protect before \footnote and \glslink 233
\@glossarysection: numbered sections and auto label added 39	\SetFootnoteAcronymStyle: Added \protect before \footnote and \glslink 239
\@gls@tmpb: changed \toksdef to \newtoks 111	symbolplural: new 61
\@gls@toc: numberline added 40	
\@p@glossarysection: numbered sections and auto label added 39	
General: amsgen now loaded (\new@ifnextchar needed) 4	
translate: translate option added 23	
\setglossarysection: new 38	
numberedsection: numberedsection package option added 7	

1.13 (2008-05-10)	
General: fixed bug that ignored 3rd parameter	124–131
\ACRfullpl: new	214
\Acrfullpl: new	214
\acrfullpl: new	213
\acrpluralsuffix: New	211
\gls@defglossaryentry: Changed default first value	77
Changed default firstplural value	77
Removed restriction on only using \newglossaryentry in the preamble	83
\newacronym: Removed restriction on only using \newacronym in the preamble	211
1.14 (2008-06-17)	
\@gls@hypergroup: new	262
General: added nonumberlist key to \printglossary	197
added numberedsection key to \printglossary	195
\firstacronymfont: new	214
\glsautoprefix: new	7
\glsnavhyperlink: changed \edef to \protected@edef	261
\glsnavhypertarget: added write to aux file	261
\glsnavigation: changed to only use labels for groups that are present ..	262
1.15 (2008-08-15)	
\@gls@link: added \glslabel	107
\gls@defglossaryentry: check for \@glo@first in description	81
check for \@glo@text in symbol	82
\gls@hypergroup: new	262
\glsnavhypertarget: added check if rerun required	261
\glssettoctitle: new	31
\printglossary: changed the way the TOC title is set	181
1.16 (2008-08-27)	
\@GLS@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	120
\@GLSpl: Test glossary type is \acronymtype in addition to checking if footnote option has been used	123
\@GLS@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	120
\@GLSpl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	122
\@GLS@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	119
\@GLSdisp: Test glossary type is \acronymtype in addition to checking if footnote option has been used	123
\@GLSpl@: Test glossary type is \acronymtype in addition to checking if footnote option has been used	121
\@GLStarget: raised the hypertarget so the target text doesn't scroll off the top of the page	117
\gls@defglossaryentry: Changed def to let	77
1.17 (2008-12-26)	
\@do@wrglossary: new	176
\@do@seeglossary: new	179
\@glo@storeentry: new	84
\@gls@glossary: changed definition to use \index instead of \@index	174
\@glsdefaultplural: new	65
\@glsdefaultsort: new	65
\@gls@hypernumber: new	208
\@glsnoname: new	64
\@glsnonextpages: new	198
General: added xindy support	25
parent: new	62
see: new	62
\gls@defglossaryentry: added nonumberlist key	78
added parent key	78
added see key	78
Stored main part of entry format when entry is defined	82
\gls@suffixF: new	35
\gls@suffixFF: new	36
\gls@wrglossary: modified to allow for xindy support	174

\glshyperlink: new	152	\SetDescriptionFootnoteAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	233
\glshypernumber: modified to allow material to be attached to location	208	\SetFootnoteAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	239
\glsnahyperlink: replaced \hyperlink to \@glslink	261	\SetSmallAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	242
\glsnahypertarget: replaced \hypertarget to \@glstarget ...	261	2.01 (2009 May 30) \@glsl@link: moved \@do@wrglossary before term is displayed to prevent unwanted whatsit	108
\glsee: new	180	\forall glossaries: replaced \ifthenelse with \ifx	49
\glseeformat: new	180	\forall glossentries: replaced \ifthenelse with \ifx	49
\glSetSuffixF: new	36	\gldefmain: new	13
\glSetSuffixFF: new	36	\gldescwidth: changed \linewidth to \hsize	269, 291
\ifglxindy: new	25	\glslistdottedwidth: changed \linewidth to \hsize	269
\istfilename: added xindy support ...	34	\glspagelistwidth: changed \linewidth to \hsize	269, 291
\newglossarystyle: made \newglossarystyle long	207	nomain: added nomain package option	13
\nopostdesc: new	34	\writeist: removed item_02 - no such makeindex key	160
nonumberlist: new	62	2.02 (2007-07-13) \@printglossary: suppressed warning globally rather than locally	184
\printglossary: added check to determine if \printglossary is already defined	181	2.02 (2009-07-13) \glossarysection: changed \@mkboth to \glossarymark	37
added print language to aux file	181	\glsglossarymark: New	38
order: order package option added ...	25	2.03 (2009-09-23) \@GLS@: Added check for hyperfirst ...	120
\writeist: added xindy support	155	\@GLSpl: Added check for hyperfirst ...	123
1.18 (2009-01-14) \@glsl@loadlist: new	9	\@GLS@: Added check for hyperfirst ...	120
\@glsl@loadlong: new	8	\@GLspl@: Added check for hyperfirst ..	122
\@glsl@loadsuper: new	9	\@glsl@: Added check for hyperfirst ...	119
\@glsl@loadtree: new	9	\@glsl@link: new	106
\glsl@defglossaryentry: Changed default value of sort to \@glsldefaultsort	78	\@glsl@link: added \leavevmode ...	107
moved sort sanitization to \newglossaryentry	82	Moved entry existence check to avoid duplicate code	107
\glstarget: new	201	\@glsl@disp: Added check for hyperfirst	123
\oldacronym: new	210	\@glspl@: Added check for hyperfirst ..	121
nolist: new	9	\glsglossarymark: Added check to see if it's already defined	38
nolong: new	8	hyperfirst: new	24
sort: moved sanitization to \newglossaryentry	60		
nostyles: new	9		
nosuper: new	9		
notree: new	9		
1.19 (2009-03-02) \glsclearpage: new	40		
\glsl@disp: new	123		
\SetDescriptionAcronymStyle: changed \acronymfont to use \textsmaller instead of \smaller	237		

2.04 (2009-11-10)	
\@GLS@: Changed test to check if glossary type has been identified as a list of acronyms	120
\@GLSpl: Changed test to check if glossary type has been identified as a list of acronyms	123
\@Gls@: Changed test to check if glossary type has been identified as a list of acronyms	120
\@Glspl@: Changed test to check if glossary type has been identified as a list of acronyms	122
\@glossaryentryfield: new	83
\@glossarysubentryfield: new	83
\@gls@: Changed test to check if glossary type has been identified as a list of acronyms	119
\@glsacronymlists: new	15
\@glldisp: Changed test to check if glossary type has been identified as a list of acronyms	123
\@glspl@: Changed test to check if glossary type has been identified as a list of acronyms	121
\@newglossaryentryposthook: new ..	83
\@newglossaryentryprehook: new ..	83
acronymlists: new	16
\DeclareAcronymList: new	15
\DefineAcronymSynonyms: new	228
\gls@defglossaryentry: added user1-6 keys	78
\glsadd: fixed bug that ignored counter	153
\Glsentryuseri: new	148
\glsentryuseri: new	148
\Glsentryuserii: new	149
\glsentryuserii: new	149
\Glsentryuseriii: new	149
\glsentryuseriii: new	149
\Glsentryuseriv: new	149
\glsentryuseriv: new	149
\Glsentryuserv: new	149
\glsentryuserv: new	149
\Glsentryuservi: new	150
\glsentryuservi: new	150
\ns@newglossary: added check to determine if \gls@<type>@display and \gls@<type>@displayfirst have been defined.	57
\SetAcronymLists: new	16
\SetDefaultAcronymDisplayStyle: new	229
\SetDefaultAcronymStyle: new	230
\SetDescriptionAcronymDisplayStyle: new	235
\SetDescriptionDUAAcronymDisplayStyle: new	233
\SetDescriptionFootnoteAcronymDisplayStyle: new	231
\SetDUADisplayStyle: new	243
\SetFootnoteAcronymDisplayStyle: new	237
\SetSmallAcronymDisplayStyle: new	240
2.05 (2010-02-06)	
\@glldisp: Added closing brace. Patch provided by Sergiu Dotenco	123
Removed spurious brace. Patch provided by Sergiu Dotenco	124
\writeist: Added \string before opening and closing braces. Patch provided by Segiu Dotenco	160
2.06 (2010-06-14)	
\altnewglossary: new	58
\CustomAcronymFields: new	245
\CustomNewAcronymDef: new	245
\SetCustomDisplayStyle: new	245
\SetCustomStyle: new	246
2.07 (2010-07-10)	
General: glssadd format key stored in \@glsnumberformat (was mistakenly stored in \@glo@format)	153
3.0 (2010-07-12)	
\@makeglossary: Added check for savewrites	165
\gls@wrglossary: modified to take into account savewrites	174
3.0 (2010/03/31)	
\@set@glo@numformat: added 4th argument	109
3.0 (2011-04-02)	
\@do@wrglossary: added check for hyper location prefix	177
modified to use new format	176
\@@glossarysec: replaced \@ifundefined with \ifcsundef ...	6
\@do@seeglossary: Sanitize and escape cross-referencing information	179
\@gls@counterwithin: new	10

\@gls@ifinlist: new	41	\glsadd: added	
\@gls@link: added		\@gls@saveentrycounter	153
\@gls@saveentrycounter	108	\GlsAddXdyCounters: new	41
added \@gls@setsort	108	\glentrycounterlabel: new	200
\@gls@saveentrycounter: new	108	\glentryitem: new	200
\@gls@setupsort@def: new	11	\Glentrylong: new	150
\@gls@setupsort@standard: new	11	\glentrylong: new	150
\@gls@setupsort@use: new	12	\Glentrylongpl: new	150
\@gls@xdy@locationlist: new	44	\glentrylongpl: new	150
\@glslink: replaced \@ifundefined		\Glentryshort: new	150
with \ifcsundef	117	\glentryshort: new	150
\@glsnextpages: new	198	\Glentryshortpl: new	150
\@print@glossary: replaced		\glentryshortpl: new	150
\@ifundefined with \ifcsundef ..	184	\glsgetgrouptitle: replaced	
\@printglossary: added		\@ifundefined with \ifcsundef ..	204
\currentglossary	183	\gls glossarymark: replaced	
added \glsnextpages	183	\@ifundefined with \ifcsundef ..	38
make toctitle default to title	183	\glshyperlink: changed default from	
\@xdyattributelist: new	41	\glentryname to \glentrytext ..	152
General: added prefix to hyperlink	209	\glshypernumber: replaced	
etoolbox now loaded	4	\@ifundefined with \ifcsundef ..	208
replaced \@ifundefined with		\glsnumberformat: replaced	
\ifcsundef	29, 32, 104, 195	\@ifundefined with \ifcsundef ..	36
\acrfootnote: new	231	\glsrefentry: new	200
\ACRfull: added starred version	213	\glsresetsubentrycounter: new ...	199
\Acrfull: added starred version	212	\glsseeitem: hyperlink uses	
\acrfull: added starred version	212	\glsseeitemformat instead of	
\ACRfullpl: added starred version ...	214	\glentryname	181
\Acrfullpl: added starred version ...	214	\glsseeitemformat: new	181
\acrfullpl: added starred version ...	213	\glssortnumberfmt: new	11
\acrlinkfootnote: new	231	\glsstepentry: new	199
\acrlinkfootnote: new	231	\glsstepsubentry: new	199
savewrites: new	27	\glssubentrycounterlabel: new ...	200
see: added \@glo@seeautonumberlist ..	62	\glssubentryitem: new	200
seeautonumberlist: new	8	theglossary: replaced \@ifundefined	
\glossarysection: replaced		with \ifcsundef	200
\@ifundefined with \ifcsundef ..	37	short: new	64
\glossarystyle: replaced		shortplural: new	64
\@ifundefined with \ifcsundef ..	206	\ifglossaryexists: replaced	
\gls@codepage: replaced		\@ifundefined with \ifcsundef ..	50
\@ifundefined with \ifcsundef ..	26	\ifglentryexists: replaced	
\gls@defglossaryentry: added		\@ifundefined with \ifcsundef ..	51
\@gls@defsort	82	\istfile: deprecated	172
added short and long keys	78	glossaryentry: new	198
replaced \@ifundefined with		glossarysubentry: new	199
\ifcsundef	78	\newglossaryentry: replaced	
\gls@doclearpage: replaced		\DeclareRobustCommand with	
\@ifundefined with \ifcsundef ..	39	\newrobustcmd	67

\newglossarystyle: replaced		\showglouseriii: new	248
\@ifundefined with \ifcsundef .	207	\showglouseriv: new	249
\ns@newglossary: added		\showglouserv: new	249
\@gls@defsortcount	58	\showglouservi: new	249
replaced \@ifundefined with		subentrycounter: new	10
\ifcsundef	57	\writeist: added xindy-only macro	
entrycounter: new	10	definitions to glossary open tag	157
entrycounterwithin: new	10	modified to support new format	155
\oldacronym: replaced \@ifundefined		3.01 (2011-04-12)	
with \ifcsundef	210	\@glswritefiles: added check for	
compatible-2.07: compatible-2.07		empty glossaries	173
option added	27	General: made robust	120
long: new	64	\ACRfull: made robust	213
longplural: new	64	\Acrfull: made robust	212
nonumberlist: now boolean	62	\acrfull: made robust	212
sort: new	10	\acrfullformat: removed	
counter: replaced \@ifundefined with		\acronymfont as it should already be	
\ifcsundef	61	set in the second argument.	212
\printglossary: replaced		\ACRfullpl: made robust	214
\@ifundefined with \ifcsundef .	181	\Acrfullpl: made robust	214
\SetDescriptionFootnoteAcronymDisplayStyle		\acrfullpl: made robust	213
expanded options link options	231	\ACRlong: made robust	141
\setentrycounter: added optional		\Acrlong: made robust	140
argument	205	\acrlong: made robust	140
\showacronymlists: new	251	\ACRlongpl: made robust	143
\showglocounter: new	248	\Acrlongpl: made robust	142
\showgloDESC: new	249	\acrlongpl: made robust	142
\showgloDESCplural: new	249	\ACRshort: made robust	137
\showglofirst: new	247	\Acrshort: made robust	137
\showglofirstpl: new	248	\acrshort: made robust	136
\showgloflag: new	251	\ACRshortpl: made robust	139
\showgloindex: new	250	\Acrshortpl: made robust	139
\showglolevel: new	247	\acrshortpl: made robust	138
\showglongame: new	249	\Gls: made robust	119
\showgloparent: new	247	\glsadd: made robust	153
\showgloplural: new	247	\glsaddall: made robust	153
\showglosort: new	250	\GLSdesc: made robust	129
\showglossaries: new	251	\Glsdesc: made robust	128
\showglossarycounter: new	252	\glsdesc: made robust	128
\showglossaryentries: new	252	\GLSdescplural: made robust	130
\showglossaryin: new	251	\Glsdescplural: made robust	129
\showglossaryout: new	252	\glsdescplural: made robust	129
\showglossarytitle: new	252	\glsfirst: made robust	125
\showglosymbol: new	250	\GLSfirstplural: made robust	127
\showglosymbolplural: new	250	\Glsfirstplural: made robust	127
\showglotext: new	247	\glsfirstplural: made robust	127
\showglotype: new	248	\glslink: made robust	106
\showglouserI: new	248	\GLSname: made robust	128
\showglouserII: new	248	\Glsname: made robust	128

\glsname: made robust	127	\@printglossary: add a way to fetch	
\GLSpl: made robust	122	current entry label	184
\Glspl: made robust	121	savenumberlist: new	8
\glspl: made robust	121	ucmark: new	10
\GLSplural: made robust	126	\gls@defglossaryentry: added	
\GLSsymbol: made robust	130	numberlist element	81
\Glsymbol: made robust	130	\gls@save@numberlist: new	181
\glssymbol: made robust	130	\gls@wrglossary: added check for	
\GLSsymbolplural: made robust	131	glossary file defined	174
\Glsymbolplural: made robust	131	\glsdisplaynumberlist: new	151
\glssymbolplural: made robust	131	\glsentrycounter: set default value ..	108
\Glstext: made robust	125	\Glsentryfull: fixed bug (replaced	
\glstext: made robust	124	\glsentryshortpl with	
\GLSuseri: made robust	132	\glsentryshort)	151
\Glsuseri: made robust	132	\glsentryfullpl: fixed bug (replaced	
\glsuseri: made robust	132	\glsentryshort with	
\GLSuserii: made robust	133	\glsentryshortpl)	151
\Glsuserii: made robust	133	\glsentrynumberlist: new	151
\glsuserii: made robust	132	\glsmoveentry: new	83
\GLSuseriii: made robust	134	\glsresetsubentrycounter: new ...	199
\Glsuseriii: made robust	133	\ifglshaschildren: new	52
\glsuseriii: made robust	133	\ifglshasparent: new	53
\GLSuseriv: made robust	134	\makeglossaries: added list parser ..	168
\Glsuseriv: made robust	134	indexonlyfirst: new	24
\glsuseriv: made robust	134	\renewglossarystyle: new	207
\GLSuserv: made robust	135	\showglossaryentries: fixed misspelt	
\Glsuserv: made robust	135	command	252
\glsuserv: made robust	135	\SmallNewAcronymDef: fixed broken	
\GLSuservi: made robust	136	short and long plural	241
\Glsuservi: made robust	136	3.03 (2012/09/21)	
\glsuservi: made robust	136	\@gls@sanitizesort: new	18
3.02 (2012-05-19)		\@gls@setupsort@standard: used	
\glsnumlistlastsep: new	152	\@gls@sanitizesort	11
\glsnumlistsep: new	152	\@printglossary: allow title to override	
3.02 (2012-05-21)		default toctitle	183
\@do@wrglossary: changed		General: allow title to set toctitle	195
\@glslocref to		\glsinlinedescformat: new	265
\theglsentrycounter	178	\glsinlineemptydescformat: new ..	265
\@do@wrglossary: changed		\glsinlinenameformat: new	265
\@do@wr@glossary to test for		\glsinlinepostchild: new	265
indexonlyfirst option; put old		\glsinlinesubdescformat: new	265
\@do@wr@glossary code into		\glsinlinesubnameformat: new	265
\@do@wrglossary	175	\glspostinline: replaced “.” with	
\@gls@missingnumberlist: new	65	\glspostdescription	265
\@glswritefiles: added check for		list: added check for glsnogroupskip ..	266
existence of token in case		altlongragged4col: added check for	
\makeglossaries has been		glsnogroupskip	284
omitted	173	altsuperragged4col: added check for	
		glsnogroupskip	302

alttree: added check for		\gls@disablepagerefexpansion: new	175
glsnogroupskip	312	\gls@numberpage: new	175
index: added check for glsnogroupskip	306	\gls@protected@pagefmts: new	175
nogroupskip: new	9	\gls@romanpage: new	176
long: added check for glsnogroupskip	270	\glsdefmain: added check for doc	
long3col: added check for		package	13
glsnogroupskip	271	\glsorg@endtheglossary: new	5
long4col: added check for		\glsorg@theglossary: new	5
glsnogroupskip	273	\PrintChanges: new	5
longragged: added check for		3.05 (2013-04-21)	
glsnogroupskip	281	\@do@wrglossary: add Roman case.	
longragged3col: added check for		Fixed bugs in the else statements ..	177
glsnogroupskip	282	\@gls@link: added check for	
nopostdot: new	9	“nohypertypes”	107
tree: added check for glsnogroupskip	307	mcolalttree: replaced ‘2’ with	
treenoname: added check for		\glsmcols	289
glsnogroupskip	309	mcolindex: replaced ‘2’ with \glsmcols	286
super: added check for glsnogroupskip	292	mcolindexspannav: replaced ‘2’ with	
super3col: added check for		\glsmcols	287
glsnogroupskip	294	mcoltree: replaced ‘2’ with \glsmcols	287
super4col: added check for		mcoltreenoname: replaced ‘2’ with	
glsnogroupskip	295	\glsmcols	288
superragged: added check for		mcoltreesspannav: replaced ‘2’ with	
glsnogroupskip	299	\glsmcols	288
superragged3col: added check for		\gls@protected@pagefmts: added	
glsnogroupskip	301	Roman to list	175
3.04 (2012-11-11)		\gls@Romanpage: new	176
altlist: replaced \newline with		\glsgetgrouplabel: fixed bug (typo in	
paragraph break	267	\equal)	205
3.04 (2012-11-18)		\nopostdesc: made robust	34
\@do@wrglossary: changed		3.05 (2013/04/21)	
\theglsentrycounter back to		\@gls@nohyperlist: new	16
\@glslocref	178	\GlsDeclareNoHyperList: new	16
\@do@wrglossary: modified to		nohypertypes: new	16
compensate for possible incorrect		3.06 (2013/06/17)	
page number	177	\@xdy@main@language: Changed back to	
\@gls@escbsdq: unsanitize		using \language	25
\gls@numberpage, \gls@alphpage,		\findrootlanguage: Obsoleted	48
\gls@Alphpage and		3.07 (2013-07-05)	
\gls@romanpage	110	\@gls@link: fixed bug that failed to find	
\@print@glossary: Moved aux write to		entry in list	107
end of document to prevent		\glossarypreamble: modified to work	
unwanted whatsit occurring here. ..	184	with \setglossarypreamble	37
General: Added check for doc package	4	\gls@doclearpage: added check for	
added datatool-base as a required		openright	39
package	4	\glspostdescription: Added	
added local key	104	spacefactor code	9
\gls@Alphpage: new	175	\GlsSetXdyCodePage: Added check for	
\gls@alphpage: new	175	fontspec	48

\SetDescriptionAcronymDisplayStyle: now using \glsdoparenifnotempty	235	\ifglshasdesc: new	53
\setglossarypreamble: new	37	\ifglshassymbol: new	53
3.08a (2013-08-30)		altlongragged4col: updated to use \glossentry and \subglossentry	284
list: updated list style to use \glossentry and \subglossentry	266	alttree: updated to use \glossentry and \subglossentry	310
listdotted: updated listdotted style to use \glossentry and \subglossentry	268	index: added paragraph break at end of environment	305
altlist: updated altlist style to use \glossentry and \subglossentry	267	updated to use \glossentry and \subglossentry	305
inline: updated inline style to use \glossentry and \subglossentry	264	long: updated to use \glossentry and \subglossentry	270
3.08a (2013-09-28)		longragged: updated to use \glossentry and \subglossentry	280
\@glo@storeentry: no longer need to check for special characters in any of the fields other than sort	84	longragged3col: updated to use \glossentry and \subglossentry	282
updated for \glossentry	84	tree: updated to use \glossentry and \subglossentry	307
\@glossaryentryfield: switched to \glossentry	83	\setglossarystyle: new	206
\@glossarysubentryfield: switched to \subglossentry	83	\setglossentrycompatibility: new	203
General: added nogroupskip key to \printglossary	196	superragged: updated to use \glossentry and \subglossentry	299
removed definition of \@glossaryentryfield	354	3.09a (2013-10-09)	
removed definition of \@glossarysubentryfield	354	\@gls@assign@symbolplural@field: new	18
\compatibleglossentry: new	201	\@gls@default@value: new	61
\compatiblesubglossentry: new	202	\Glsentrydesc: made robust	146
\glossaryentryfield: deprecated	203	\Glsentrydescplural: made robust	146
\Glossentrydesc: new	202	\Glsentryfirst: made robust	147
\glossentrydesc: new	202	\Glsentryfirstplural: made robust	148
\Glossentryname: new	202	\Glsentryfull: made robust	151
\glossentryname: new	201	\Glsentryfullpl: made robust	151
\Glossentrysymbol: new	202	\Glsentrylong: made robust	150
\glossentrysymbol: new	202	\Glsentrylongpl: made robust	150
\gls@assign@desc@field: new	18	\Glsentryname: made robust	145
\gls@assign@descplural@field: new	18	\Glsentryplural: made robust	147
\gls@assign@field: new	67	\Glsentryshort: made robust	150
\gls@ifnotmeasuring: new	85	\Glsentryshortpl: made robust	150
\glsaddallunused: new	154	\Glsentrysymbol: made robust	147
\glsexpandfields: new	67	\Glsentrysymbolplural: made robust	147
\glsnoexpandfields: new	67	\Glsentrytext: made robust	146
\glssee: made robust	180	\Glsentryuseri: made robust	148
\glsseeformat: made robust	180	\Glsentryuserii: made robust	149
\glsseeitem: made robust	181	\Glsentryuseriii: made robust	149
\glsseelist: made robust	180	\Glsentryuseriv: made robust	149
\ifglsdessuppressed: new	53	\Glsentryuserv: made robust	149
		\Glsentryuservi: made robust	150
		\glstextup: new	211

\ifglshassymbol: changed test to check for \@gls@default@symbol	53	\@Gls@: add \glsifplural, \glscapscase, \glscustomtext and \glsinsert	119
3.10a (2013-09-28)		change to using \glentryfmt style commands	119
\gls@assign@type@field: new	18	removed \makefirstuc (now dealt with in \glentryfmt)	120
3.10a (2013-10-13)		\@Glspl@: add \glsifplural, \glscapscase, \glscustomtext and \glsinsert	122
\@gls@keymap: new	69	change to using \glentryfmt style commands	122
\@gls@provide@newglossary: new ...	56	removed \makefirstuc (now dealt with in \glentryfmt)	122
\@gls@writedef: new	68	\@acrlong: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	353
\@glsdefaultplural: Obsolete	65	\@acrshort: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	352
\@glsnodelsc: new	64	\@gls@: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert	119
\@print@glossary: Added providecommand code to aux file ..	185	change to using \glentryfmt style commands	119
\gls@defglossaryentry: Changed to using \@gls@default@value	77, 78	\@gls@noexpand@fields: Fixed bug expand replaced with noexpand	66
new	77	\@glsdisp: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert	123
\gls.writedefhook: new	76	change to using \glentryfmt style commands	123
\makeglossaries: Added providecommand code to aux file ..	167	\@glspl@: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert	121
\new@glossaryentry: new	68	change to using \glentryfmt style commands	121
\ns@newglossary: added \@gls@provide@newglossary	57	General: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	137–143
3.11a (2013-10-15)		changed to just use \Glsentrydescplural	129
\@ACRlong: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	354	changed to just use \glentrydescplural	129, 130
\@ACRshort: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	352	changed to just use \Glsentrydesc ..	129
\@Acrlong: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	353		
\@Acrshort: added \glslabel, \glsifplural, \glscapscase, \glsinsert and \glscustomtext	352		
\@GLS@: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert	120		
change to using \glentryfmt style commands	120		
removed \MakeUppercase (now moved to \glentryfmt)	120		
\@GLSpl: add \glslabel, \glsifplural, \glscapscase, \glscustomtext and \glsinsert	123		
change to using \glentryfmt style commands	123		
removed \MakeUppercase as now dealt with in \glentryfmt	123		

changed to just use		
\Glsentryfirstplural	127	
changed to just use		
\glentryfirstplural	127	
changed to just use \Glsentryfirst	125	
changed to just use		
\glentryfirst	125, 126	
changed to just use \Glsentryname	128	
changed to just use \glentryname	128	
changed to just use \Glsentryplural	126	
changed to just use \glentryplural	126	
changed to just use		
\Glsentrysymbolplural	131	
changed to just use		
\glentrysymbolplural	131	
changed to just use \Glsentrysymbol	130	
changed to just use		
\glentrysymbol	130, 131	
Changed to just use \Glsentrytext	125	
changed to just use \glentrytext	124	
changed to just use		
\Glsentryuseriii	134	
changed to just use		
\glentryuseriii	133, 134	
changed to just use \Glsentryuserii	133	
changed to just use		
\glentryuserii	132, 133	
changed to just use \Glsentryuseriv	134	
changed to just use		
\glentryuseriv	134, 135	
changed to just use \Glsentryuseri	132	
changed to just use \glentryuseri	132	
changed to just use \Glsentryuservi	136	
changed to just use \glentryuservi	136	
changed to just use \Glsentryuserv	135	
changed to just use \glentryuserv	135	
Now requires textcase	4	
acronymlists: replaced		
\@addtoacronymlists with		
\DeclareAcronymList	16	
\defglsgdisplay: obsoleted	103	
\defglsgdisplayfirst: obsoleted	103	
\defglsgentryfmt: new	56	
\forglsgentries: replaced \ifx with		
\ifdefempty	49	
\gls@assign@desc: new	76	
\gls@defglossaryentry: Fixed default		
counter if none supplied	81	
\gls@doentryfmt: new	56	
\glsgdisplay: obsoleted	103	
\glsgdisplayfirst: obsoleted	102	
\glsgenentryfmt: new	97	
\glsggetgrouptitle: Added check in		
case non-Latin alphabet in use	204	
\glsglossarymark: replaced		
\MakeUppercase with		
\mfirstucMakeUppercase	38	
\glsgnavigation: switched to using		
\@gls@getgrouptitle	263	
\ifglsghasdesc: replaced \ifdefempty		
with \ifcsempy	53	
\ifglsghaslong: new	53	
\ifglsghasshort: new	54	
\ifglsghasymbol: replaced		
\ifdefempty with \ifcsempy	53	
\ifglsgused: replaced \ifthenelse with		
\ifbool	51	
\longnewglossaryentry: new	76	
\ns@newglossary: replaced		
\glsgdisplay and		
\glsgdisplayfirst with		
\glsgentryfmt	57	
compatible-3.07: cnew	27	
\SetCustomDisplayStyle: updated to		
use \defglsgentryfmt	245	
\SetDefaultAcronymDisplayStyle:		
changed to use \defglsgentryfmt	229	
\SetDescriptionAcronymDisplayStyle:		
updated to use \defglsgentryfmt	235	
\SetDescriptionDUAAcronymDisplayStyle:		
updated to use \defglsgentryfmt	233	
\SetDescriptionFootnoteAcronymDisplayStyle:		
updated to use \defglsgentryfmt	231	
\SetDUADisplayStyle: updated to use		
\defglsgentryfmt	243	
\SetFootnoteAcronymDisplayStyle:		
updated to use \defglsgentryfmt	237	
\SetSmallAcronymDisplayStyle:		
updated to use \defglsgentryfmt	240	
\setupglossaries: new	29	
\showglolong: new	250	
\showgloshort: new	250	
numbers: new	28	
symbols: new	27	
3.12a (2013-10-16)		
\gls@defglossaryentry: added		
\glsglabel	77	
\glsgaddkey: new	71	

3.13a (2013-11-05)		long: switched to \tabularnewline ..	270
\@gls@assign@symbol@field: changed		long3col: switched to	
to use \glssetnoexpandfield	18	\tabularnewline	271
\@gls@assign@symbolplural@field:		long3colheader: switched to	
changed to use		\tabularnewline	272
\glssetnoexpandfield	18	long3colheaderborder: switched to	
\@gls@link: removed \relax	108	\tabularnewline	272
\@gls@notranslatorhook: new	22	long4col: switched to	
\@gls@setupsort@standard: moved		\tabularnewline	273
\@gls@santizesort to		long4colheader: switched to	
\glsprestandardsort	11	\tabularnewline	273
ucmark: added check for memoir	10	longheader: switched to	
see: added \gls@checkseeallowed ...	62	\tabularnewline	270
\glossarysection: changed		longheaderborder: switched to	
\glossarymark to		\tabularnewline	271
\gls glossarymark	38	\SetFootnoteAcronymDisplayStyle:	
\glossarystyle: fixed bug caused by		fixed missing argument bug	238
using \ifdef instead of \ifcsdef .	206	super: switched to \tabularnewline .	292
\gls@assign@desc@field: changed to		super3col: switched to	
use \glssetnoexpandfield	18	\tabularnewline	293
\gls@assign@descplural@field:		super3colheader: switched to	
changed to use		\tabularnewline	294
\glssetnoexpandfield	18	super4col: switched to	
\gls@assign@name@field: changed to		\tabularnewline	295
use \glssetnoexpandfield	18	super4colheader: switched to	
\gls@assign@type@field: changed to		\tabularnewline	296
use \glssetexpandfield	18	super4colheaderborder: switched to	
\gls@checkseeallowed: new	62	\tabularnewline	296
\glsaddallunused: set default to		superheader: switched to	
\@glo@types	154	\tabularnewline	293
\Glsentryfull: changed to use		superheaderborder: switched to	
\acrfullformat	151	\tabularnewline	293
\Glsentryfull: changed to use			
\acrfullformat	151	3.14a (2013-11-12)	
\Glsentryfullpl: changed to use		\@glswritefiles: renamed	
\acrfullformat	151	\glswritefiles to	
\Glsentryfullpl: changed to use		\@glswritefiles and used	
\acrfullformat	151	“savewrites” option to set	
\gls glossarymark: renamed		\glswritefiles	173
\gls glossarymark to		General: new	254
\gls glossarymark to avoid conflict		acronyms: new	14
with memoir	38	\gls@def glossaryentry: added check	
\glsprestandardsort: new	10	for existence of default glossary	78
\glssetexpandfield: new	17	set the default for firstplural to be the	
\glssetnoexpandfield: new	18	value of plural	80
altsuper4colheader: switched to		xindy gloss: new	26
\tabularnewline	297	\longprovideglossaryentry: new ...	77
altsuper4colheaderborder: switched		compatible-2.07: added check for 2.07	
to \tabularnewline	297	before setting 3.07 compatibility	27
		notranslate: new	22

\provideglossaryentry:new	67	index:new	28
4.0 (2013-11-14)		\newacronymstyle:new	217
\gls@defglossaryentry: added check		long-sc-short:new	220
for first key	80	long-sc-short-desc:new	221
super: fixed typo in \subglossentry		long-short:new	218
(\glossentrydesc)	292	long-short-desc:new	221
4.01 (2013-11-16)		long-sm-short:new	220
General: fixed non-value options so that		long-sm-short-desc:new	222
they can be passed to document class .	8	long-sp-short-desc:new	221
\CustomAcronymFields: inserted		footnote:new	225
missing comma	245	footnote-desc:new	227
4.02 (2013-12-05)		footnote-sc:new	226
\@acrfull: now using \acrfullfmt ..	212	footnote-sc-desc:new	227
\@gls@indexdef:new	28	footnote-sm:new	227
\@gls@numbersdef:new	28	footnote-sm-desc:new	227
\@gls@symbolsdef:new	27	\setacronymstyle:new	217
General: Removed \acronymfont .	140–144	\SetDescriptionAcronymDisplayStyle:	
\ACRfullfmt:new	213	Moved check for empty custom text to	
\Acrfullfmt:new	213	prevent unwanted parenthetical	
\acrfullfmt:new	212	material	235
\ACRfullplfmt:new	214	\SetDescriptionFootnoteAcronymDisplayStyle:	
\Acrfullplfmt:new	214	Moved check for empty custom text to	
\acrfullplfmt:new	213	prevent unwanted parenthetical	
\acronymentry:new	216	material	231
sanitize: fixed bug that caused an error		\SetFootnoteAcronymDisplayStyle:	
here	21	Moved check for empty custom text to	
sc-short-long:new	220	prevent unwanted parenthetical	
sc-short-long-desc:new	222	material	237
\Genacrfullformat:new	102	\SetGenericNewAcronym:new	215
\genacrfullformat:new	102	\SetSmallAcronymDisplayStyle:	
\GenericAcronymFields:new	216	Moved check for empty custom text to	
\Genplacrfullformat:new	102	prevent unwanted parenthetical	
\genplacrfullformat:new	102	material	240
\Glsentryfull: bug fix: added missing		dua:new	223
\acronymfont	151	dua-desc:new	225
\glsentryfull: bug fix: added missing		numberedsection: added nameref	
\acronymfont	151	option	7
\Glsentryfullpl: bug fix: added		4.02 (2013-13-05)	
missing \acronymfont	151	\makeglossaries: made preamble only	168
\glsentryfullpl: bug fix: added		4.03 (2014-01-17)	
missing \acronymfont	151	General: changed default to \@empty	
\glsgenacfmt:new	100	instead of \relax	27
\GlsUseAcrEntryDisplayStyle:new ...	218	4.03 (2014-01-20)	
\GlsUseAcrStyleDefs:new	218	\@do@wrglossary: added	
short-long:new	219	\glsdetoklabel	178
short-long-desc:new	222	\@ACRlong: removed \glslabel	
xindynoglsnumbers:new	26	(defined in \@gls@link)	354
sm-short-long:new	220	\@ACRshort: removed \glslabel	
sm-short-long-desc:new	222	(defined in \@gls@link)	352

\@Acrlong: removed \glslabel (defined in \@gls@link)	353	\genplacrfullformat: redefined to use accessibility information	351
\@Acrshort: removed \glslabel (defined in \@gls@link)	352	\glossentryname: added \glsdetoklabel	201
\@GLS@: removed \glslabel (defined in \@gls@link)	120	\gls@defglossaryentry: added \glsdetoklabel	77
\@GLSpl: removed \glslabel (defined in \@gls@link)	123	replaced #1 with \@gls@label	78
\@Gls@: removed \glslabel (defined in \@gls@link)	119	replaced \ifthenelse with \ifdefequal	79
\@Gls@entry@field: new	144	\glsadd: added \glsdetoklabel	153
\@Glspl@: removed \glslabel (defined in \@gls@link)	122	\glsaddkey: switched to using \@gls@field@link	72
\@acrlong: removed \glslabel (defined in \@gls@link)	353	\glsdetoklabel: new	50
\@acrshort: removed \glslabel (defined in \@gls@link)	352	\glsdisplaynumberlist: added \glsdetoklabel	152
\@gls@: removed \glslabel (defined in \@gls@link)	119	\glsdoifexistsorwarn: new	51
\@gls@access@display: new	340	\glsentryaccess: switched to using \@gls@entry@field	338
\@gls@entry@field: new	144	\glsentrydescaccess: switched to using \@gls@entry@field	339
\@gls@fetchfield: new	69	\glsentrydescpluralaccess: switched to using \@gls@entry@field	339
\@gls@field@link: new	124	\glsentryfirstaccess: switched to using \@gls@entry@field	339
\@gls@link: added \glsdetoklabel .	107	\glsentryfirstplural: added \glsdetoklabel	148
moved \@gls@link@opts and \@gls@link@label to \@gls@link	107	\glsentrylongaccess: switched to using \@gls@entry@field	340
\@gls@writedef: added \glsdetoklabel	68	\glsentrylongpluralaccess: switched to using \@gls@entry@field	340
\@glsdisp: removed \glslabel (defined in \@gls@link)	123	\glsentrypluralaccess: switched to using \@gls@entry@field	339
\@glspl@: removed \glslabel (defined in \@gls@link)	121	\glsentryshortaccess: switched to using \@gls@entry@field	339
\@printglossary: added \glsdetoklabel	184	\glsentryshortpluralaccess: switched to using \@gls@entry@field	339
General: removed \glslabel (defined in \@gls@link)	137	\glsentrysymbolaccess: switched to using \@gls@entry@field	339
sc-short-long-desc: redefined to use accessibility information	358	\glsentrysymbolpluralaccess: switched to using \@gls@entry@field	339
\compatibleglossentry: added \glsdetoklabel	334	\glsentrytextaccess: switched to using \@gls@entry@field	339
\compatiblesubglossentry: added \glsdetoklabel	335	\glsngenacfmt: redefined to use accessibility information	349
\Genacrfullformat: redefined to use accessibility information	351	\glsgenentryfmt: redefined to use accessibility information	346
\genacrfullformat: redefined to use accessibility information	351		
\Genplacrfullformat: redefined to use accessibility information	351		

<code>\glshyperlink</code> : added			
<code>\glsdetoklabel</code>	152	<code>long-sc-short-desc</code> : redefined to use	
<code>\glslocalreset</code> : added		accessibility information	357
<code>\glsdetoklabel</code>	86	<code>long-short</code> : redefined to use	
<code>\glslocalunset</code> : added		accessibility information	355
<code>\glsdetoklabel</code>	86	<code>long-short-desc</code> : redefined to use	
<code>\glsmoveentry</code> : added		accessibility information	357
<code>\glsdetoklabel</code>	83	<code>long-sm-short-desc</code> : redefined to use	
replaced <code>\ifthenelse</code> with		accessibility information	357
<code>\ifdefequal</code>	83	<code>footnote</code> : redefined to use accessibility	
<code>\glsrefentry</code> : added <code>\glsdetoklabel</code>	200	information	361
<code>\glsreset</code> : added <code>\glsdetoklabel</code> ...	86	<code>footnote-desc</code> : redefined to use	
<code>\glsseelist</code> : added <code>\expandafter</code>		accessibility information	363
<code>commands</code>	180	<code>footnote-sc</code> : redefined to use	
<code>\glsstepentry</code> : added		accessibility information	363
<code>\glsdetoklabel</code>	199	<code>footnote-sc-desc</code> : redefined to use	
<code>\glsstepsubentry</code> : added		accessibility information	364
<code>\glsdetoklabel</code>	199	<code>footnote-sm</code> : redefined to use	
<code>\glsunset</code> : added <code>\glsdetoklabel</code> ...	86	accessibility information	363
<code>short-long</code> : commented spurious EOL	220	<code>footnote-sm-desc</code> : redefined to use	
redefined to use accessibility		accessibility information	364
information	356	<code>\renewacronymstyle</code> : new	217
<code>short-long-desc</code> : redefined to use		<code>\showglocounter</code> : added	
accessibility information	358	<code>\glsdetoklabel</code>	248
<code>\ifglshdescsuppressed</code> : added		<code>\showgloDESC</code> : added <code>\glsdetoklabel</code>	249
<code>\glsdetoklabel</code>	53	<code>\showgloDESCaccess</code> : added	
fixed typo	53	<code>\glsdetoklabel</code>	370
<code>\ifglshentryexists</code> : added		<code>\showgloDESCplural</code> : added	
<code>\glsdetoklabel</code>	51	<code>\glsdetoklabel</code>	250
<code>\ifglshaschildren</code> : added		<code>\showgloDESCpluralaccess</code> : added	
<code>\glsdetoklabel</code>	52	<code>\glsdetoklabel</code>	370
<code>\ifglshasdesc</code> : added		<code>\showglofirst</code> : added	
<code>\glsdetoklabel</code>	53	<code>\glsdetoklabel</code>	248
<code>\ifglshasfield</code> : new	54	<code>\showglofirstaccess</code> : added	
<code>\ifglshaslong</code> : added		<code>\glsdetoklabel</code>	370
<code>\glsdetoklabel</code>	53	<code>\showglofirstpl</code> : added	
<code>\ifglshasparent</code> : added		<code>\glsdetoklabel</code>	248
<code>\glsdetoklabel</code>	53	<code>\showglofirstpluralaccess</code> : added	
<code>\ifglshasshort</code> : added		<code>\glsdetoklabel</code>	370
<code>\glsdetoklabel</code>	54	<code>\showgloflag</code> : added <code>\glsdetoklabel</code>	251
<code>\ifglshassymbol</code> : added		<code>\showgloindex</code> : added	
<code>\glsdetoklabel</code>	53	<code>\glsdetoklabel</code>	251
replaced <code>\ifcempty</code> with		<code>\showglolevel</code> : added	
<code>\ifdefempty</code> and replaced <code>\ifx</code> with		<code>\glsdetoklabel</code>	247
<code>\ifdefequal</code>	53	<code>\showgloLONG</code> : added <code>\glsdetoklabel</code>	250
<code>\ifglshused</code> : added <code>\glsdetoklabel</code> ..	51	<code>\showgloLONGaccess</code> : added	
<code>sm-short-long-desc</code> : redefined to use		<code>\glsdetoklabel</code>	371
accessibility information	358	<code>\showgloLONGpluralaccess</code> : added	
		<code>\glsdetoklabel</code>	371
		<code>\showgloNAME</code> : added <code>\glsdetoklabel</code>	249

\showglongnameaccess: added	4.04 (2014-03-04)	
\glsdetoklabel	370	\@gls@getcounterprefix: added
\showgloparent: added		warning if no prefix can be formed . 179
\glsdetoklabel	247	4.04 (2014-03-06)
\showgloplural: added		\@gls@noidx@nosanitizesort: new . 19
\glsdetoklabel	247	\@gls@noidx@sanitizesort: new ... 19
\showglopluralaccess: added		\@gls@nosanitizesort: new 19
\glsdetoklabel	370	\@gls@sanitizesort: new 19
\showgloshort: added		\@glo@addchildren: new 186
\glsdetoklabel	250	\@glo@do@sortentries: new 187
\showgloshortaccess: added		\@glo@grabfirst: new 192
\glsdetoklabel	371	\@glo@sortedinsert: new 187
\showgloshortpluralaccess: added		\@glo@sortentries: new 186
\glsdetoklabel	371	\@glo@sorthandler@case: new 188
\showglosort: added \glsdetoklabel	250	\@glo@sorthandler@letter: new ... 188
\showglosymbol: added		\@glo@sorthandler@nocase: new ... 188
\glsdetoklabel	250	\@glo@sorthandler@word: new 187
\showglosymbolaccess: added		\@glo@sortmacro@case: new 189
\glsdetoklabel	370	\@glo@sortmacro@def: new 190
\showglosymbolplural: added		\@glo@sortmacro@def@do: new 190
\glsdetoklabel	250	\@glo@sortmacro@letter: new 189
\showglosymbolpluralaccess: added		\@glo@sortmacro@nocase: new 190
\glsdetoklabel	370	\@glo@sortmacro@standard: new ... 189
\showglotext: added \glsdetoklabel	247	\@glo@sortmacro@use: new 190
\showglotextaccess: added		\@glo@sortmacro@word: new 188
\glsdetoklabel	370	\@gls@getothergrouptitle: new ... 205
\showglotype: added \glsdetoklabel	248	\@gls@noidx@do: new 192
\showglouser: added		\@gls@noref@warn: new 172
\glsdetoklabel	248	\@gls@reference: new 194
\showglouserii: added		\@gls@warnonglossdefined: new 17
\glsdetoklabel	248	\@gls@warnontheGLOSSdefined: new . 17
\showglouseriii: added		\@no@makeglossaries: new 172
\glsdetoklabel	249	\@print@glossary: new 184
\showglouseriv: added		\@print@noidx@glossary: new 191
\glsdetoklabel	249	\@print@gloss@setsort: new 182
\showglouserv: added		\@print@glossary: new 182
\glsdetoklabel	249	General: added sort key to printgloss
\showglouservi: added		group 197
\glsdetoklabel	249	\compatibleglossentry: changed
dua: fixed bug in \acrfullfmt	224	\newcommand to \def as is may or
fixed bug in \Acrrfullplfmt	224	may not be defined 334
fixed bug in \acrfullplfmt	224	\compatiblesubglossentry: changed
redefined to use accessibility		\newcommand to \def as is may or
information	359	may not be defined 335
dua-desc: commented spurious EOL ..	225	\defglsdisplayfirst: fixed unwanted
redefined to use accessibility		space 103
information	361	\glo@grabfirst: new 191
		\gls@defglossaryentry: replaced \ifx
		with \ifdefvoid 82

\glsnoidxdisplayloc: new	194	\@ACRshort: added	
\glsnoidxdisplayloclisthandler:		\do@gl@link@checkfirsthyper	352
new	194	\@Acrlong: added	
\glsnoidxloclist: new	193	\do@gl@link@checkfirsthyper	353
\glsnoidxloclisthandler: new	193	\@Acrshort: added	
\glsnoidxstripaccents: new	19	\do@gl@link@checkfirsthyper	352
alttree: moved hangindent and		\@GLS@: moved \glsifhyper	120
parindent assignments outside level		moved check for first use to	
test	310	\@gl@link	120
\makeglossaries: Moved definition of		\@GLSpl: moved \glsifhyper	123
\glswrite to \makeglossaries ..	166	moved check for first use to	
\makenoidxglossaries: new	169	\@gl@link	123
\printglossary: changed to use new		\@Gls@: moved \glsifhyper	119
\@printglossary	182	moved check for first use to	
\printnoidxglossaries: new	182	\@gl@link	119
\printnoidxglossary: new	182	\@Glspl@: moved \glsifhyper	122
\showgloclist: new	251	moved check for first use to	
\warn@noprintglossary: Activate		\@gl@link	122
warning in \makeglossaries	181	\@acrlong: added	
\writeist: checked for definition of		\do@gl@link@checkfirsthyper	353
\glswrite	155, 160	\@acrshort: added	
4.06 (2014-03-12)		\do@gl@link@checkfirsthyper	351
\@GLS@: added \glsifhyper	120	\@closegls: new	165
\@GLSpl: added \glsifhyper	123	\@gl@: moved \glsifhyper	119
\@Gls@: added \glsifhyper	119	moved check for first use to	
\@Glspl@: added \glsifhyper	122	\@gl@link	119
\@gl@: added \glsifhyper	119	\@gl@automake: new	165
\@gl@numbersdef: added hook to set		\@gl@doautomake: new	26
toc title	28	\@gl@field@link: added assignment	
\@gl@symbolsdef: added hook to set		of	
toc title	27	\do@gl@link@checkfirsthyper	124
\@gl@disp: added \glsifhyper	123	\@gl@forbidtexext: new	56
\@gl@spl@: added \glsifhyper	121	\@gl@hyp@opt: new	105
General: added \glsifhyper	137–144	\@gl@link: removed redundancy	107
acronym: added hook to set toc title	14	renamed \gl@type to \glstype ...	107
acronyms: added hook to set toc title ...	14	\@gl@link@checkfirsthyper: new .	106
\gl@defmain: added hook to set toc title	13	\@gl@disp: moved \glsifhyper	123
4.07 (2014-04-04)		moved check for first use to	
\@glossarysection: added optional		\@gl@link	123
argument when using unstarred		\@gl@spl@: moved \glsifhyper	121
version	39	moved check for first use to	
\@gl@noidx@do: added \global in case		\@gl@link	121
it's used in a tabular-like style	192	\@ignored@glossaries: new	59
\Acrfullplfmt: fixed no case change		General: added entrycounter option to	
bug	214	printgloss family	196
\gl@letentryfield: new	144	added nopostdot option to	
4.08 (2014-07-30)		printgloss family	196
\@ACRlong: added		added subentrycounter option to	
\do@gl@link@checkfirsthyper	353	printgloss family	196

explicitly initialise hyper key	104	removed \@sGLSuseriii	133
moved \glsifhyper	137–144	removed \@sGlsuseriii	133
removed \@sACRlongpl	143	removed \@sglsuseriii	132
removed \@sAcrlongpl	143	removed \@sGLSuseriv	135
removed \@sacrlongpl	142	removed \@sGlsuseriv	134
removed \@sACRlong	141	removed \@sglsuseriv	134
removed \@sAcrlong	141	removed \@sGLSuseri	132
removed \@sacrlong	140	removed \@sGlsuseri	132
removed \@sACRshortpl	139	removed \@sglsuseri	132
removed \@sAcrshortpl	139	removed \@sGLSuservi	136
removed \@sacrshortpl	138	removed \@sGlsuservi	136
removed \@sACRshort	138	removed \@sglsuservi	136
removed \@sAcrshort	137	removed \@sGLSuserv	135
removed \@sacrshort	136	removed \@sGlsuserv	135
removed \@sgls@link	106	removed \@sglsuserv	135
removed \@sGLSdescplural	130	removed \@sGLS	120
removed \@sGlsdescplural	129	removed \@sGls	119
removed \@sglsdescplural	129	removed \@sgls	118
removed \@sGLSdesc	129	removed \@thirdofthree (defined in	
removed \@sGlsdesc	129	kernel)	118
removed \@sglsdesc	128	removed sPGLS	259
removed \@sglsdisp	123	removed sPgls	257
removed \@sGLSfirstplural	127	removed spgls	256
removed \@sGlsfirstplural	127	removed sPGLSpl	259
removed \@sglsfirstplural	127	removed sPglspl	258
removed \@sGLSfirst	126	removed spglspl	257
removed \@sGlsfirst	125	\ACRfull: removed \s@ACRfull	213
removed \@sglsfirst	125	switched to using \@gls@hyp@opt ..	213
removed \@sGLSname	128	\Acrfull: removed \@sAcrfull	212
removed \@sGlsname	128	switched to using \@gls@hyp@opt ..	212
removed \@sglsname	127	\acrfull: removed \@sacrfull	212
removed \@sGLSplural	126	switched to using \@gls@hyp@opt ..	212
removed \@sGlsplural	126	\ACRfullpl: removed \s@ACRfullpl ..	214
removed \@sglsplural	126	switched to using \@gls@hyp@opt ..	214
removed \@sGLSpl	122	\Acrfullpl: removed \s@Acrfullpl ..	214
removed \@sGlspl	122	switched to using \@gls@hyp@opt ..	214
removed \@sglspl	121	\acrfullpl: removed \s@acrfullpl ..	213
removed \@sGLSsymbolplural	131	switched to using \@gls@hyp@opt ..	213
removed \@sGlsymbolplural	131	\ACRlong: switched to using	
removed \@sglsymbolplural	131	\@gls@hyp@opt	141
removed \@sGLSsymbol	130	\Acrlong: switched to using	
removed \@sGlsymbol	130	\@gls@hyp@opt	140
removed \@sglsymbol	130	\acrlong: switched to using	
removed \@sGLStext	124	\@gls@hyp@opt	140
removed \@sGlstext	125	\ACRlongpl: switched to using	
removed \@sglstext	124	\@gls@hyp@opt	143
removed \@sGLSuseriii	134	\Acrlongpl: switched to using	
removed \@sGlsuseriii	133	\@gls@hyp@opt	142
removed \@sglsuseriii	133		

\acrlongpl: switched to using \@gls@hyp@opt	142	\GLSfirst: switched to using \@gls@hyp@opt	125
\ACRshort: switched to using \@gls@hyp@opt	137	\Glsfirst: switched to using \@gls@hyp@opt	125
\Acrshort: switched to using \@gls@hyp@opt	137	\glsfirst: switched to using \@gls@hyp@opt	125
\acrshort: switched to using \@gls@hyp@opt	136	\GLSfirstplural: switched to using \@gls@hyp@opt	127
\ACRshortpl: switched to using \@gls@hyp@opt	139	\Glsfirstplural: switched to using \@gls@hyp@opt	127
\Acrshortpl: switched to using \@gls@hyp@opt	139	\glsfirstplural: switched to using \@gls@hyp@opt	127
\acrshortpl: switched to using \@gls@hyp@opt	138	\glsifhyper: deprecated	105
\forallacronyms: new	49	\glslink: switched to using \@gls@hyp@opt	106
\GLS: switched to using \@gls@hyp@opt	120	\glslinkcheckfirsthyperhook: new	107
\Gls: switched to using \@gls@hyp@opt	119	\glslinkvar: new	105
\gls: switched to using \@gls@hyp@opt	118	\GLSname: switched to using \@gls@hyp@opt	128
\gls@defglossaryentry: added check for ignored glossary	79	\Glsname: switched to using \@gls@hyp@opt	128
\gls@istfilebase: new	35	\glsname: switched to using \@gls@hyp@opt	127
\glsaddkey: removed \@sGLS@user@<key>	73	\GLSpl: switched to using \@gls@hyp@opt	122
removed \@sGls@user@<key>	72	\Glspl: switched to using \@gls@hyp@opt	121
removed \@sgls@user@<key>	72	\glspl: switched to using \@gls@hyp@opt	121
switched to using \@gls@hyp@opt	72, 73	\GLSplural: switched to using \@gls@hyp@opt	126
\GLSdesc: switched to using \@gls@hyp@opt	129	\Glsplural: switched to using \@gls@hyp@opt	126
\Glsdesc: switched to using \@gls@hyp@opt	128	\glsplural: switched to using \@gls@hyp@opt	126
\glsdesc: switched to using \@gls@hyp@opt	128	\glsspace: new	212
\GLSdescplural: switched to using \@gls@hyp@opt	130	\GLSsymbol: switched to using \@gls@hyp@opt	130
\Glsdescplural: switched to using \@gls@hyp@opt	129	\Glsymbol: switched to using \@gls@hyp@opt	130
\glsdescplural: switched to using \@gls@hyp@opt	129	\glssymbol: switched to using \@gls@hyp@opt	130
\glsdisablehyper: added \KV@glslink@hyperfalse to definition	118	\GLSsymbolplural: switched to using \@gls@hyp@opt	131
\glsdisp: switched to using \@gls@hyp@opt	123	\Glsymbolplural: switched to using \@gls@hyp@opt	131
\glsdohyperlink: new	117	\glssymbolplural: switched to using \@gls@hyp@opt	131
\glsdohypertarget: new	117		
\glsenablehyper: added \KV@glslink@hypertrue to definition	118		

\GLStext: switched to using \@gls@hyp@opt	124	\ns@newglossary: added \@glotype@<name>@log	57
\Glstext: switched to using \@gls@hyp@opt	125	new	57
\glstext: switched to using \@gls@hyp@opt	124	\p@gls@hyp@opt: new	105
\glstreenamefmt: new	304	\PGLS: changed to use \@gls@hyp@opt	259
\GLSuseri: switched to using \@gls@hyp@opt	132	\Pgls: changed to use \@gls@hyp@opt	257
\Glsuseri: switched to using \@gls@hyp@opt	132	\pgls: changed to use \@gls@hyp@opt	256
\glsuseri: switched to using \@gls@hyp@opt	132	\PGLSpl: changed to use \@gls@hyp@opt	259
\GLSuserii: switched to using \@gls@hyp@opt	133	\Pglspl: changed to use \@gls@hyp@opt	258
\Glsuserii: switched to using \@gls@hyp@opt	133	\pglspl: changed to use \@gls@hyp@opt	257
\glsuserii: switched to using \@gls@hyp@opt	132	\s@gls@hyp@opt: new	105
\GLSuseriii: switched to using \@gls@hyp@opt	134	\s@newglossary: new	57
\Glsuseriii: switched to using \@gls@hyp@opt	133	automake: new	26
\glsuseriii: switched to using \@gls@hyp@opt	133	4.09 (2014-08-12)	
\GLSuseriv: switched to using \@gls@hyp@opt	134	\glsaddkey: fixed bug in user commands	72
\Glsuseriv: switched to using \@gls@hyp@opt	134	4.10 (2014-08-27)	
\glsuseriv: switched to using \@gls@hyp@opt	134	\@Gls@acrentryname: new	145
\GLSuserv: switched to using \@gls@hyp@opt	135	\@Gls@entryname: new	145
\Glsuserv: switched to using \@gls@hyp@opt	135	\@gls@glossary: Renamed \@glossary to \@gls@glossary	174
\glsuserv: switched to using \@gls@hyp@opt	135	\glspercentchar: new	155
\ifignoredglossary: new	59	\glstildechar: new	155
altlongragged4col: fixed bug that displayed description instead of symbol	284	alttree: moved space after symbol	311, 312
\newglossary: added starred version ..	57	4.11 (2014-09-01)	
\newignoredglossary: new	59	\@do@wrglossary: added hook	177
		sanitize: none option	21
		\gls@wrglossary: renamed from \@wrglossary to \gls@wrglossary	174
		\glsaddprotectedpagefmt: new	176
		\glsbackslash: new	154
		4.12 (2014-11-22)	
		\@gls@addpredefinedattributes: Added glsignore attribute	43
		\@gls@adjustmode: new	153
		\@gls@notranslatorhook: removed ...	22
		\@gls@toc: added \protect to \numberline	40
		\@gls@usetranslator: new	22
		\glsacrpluralsuffix: new	31
		\glsadd: added check for vertical mode	153
		\glsaddallunused: replaced @gobble with glsignore	154
		\glsifusedtranslatordict: new	22
		\glsignore: new	154
		\glsupacrpluralsuffix: new	31
		\ProvidesGlossariesLang: new	32

\@printglossary: added		\glslistgroupheaderfmt: new	266
\@gls@preglossaryhook	184	\glslistnavigationitem: new	266
\do@glsglisablehyperinlist: new	107	\glstreegroupheaderfmt: new	304
\doifglossarynoexistsordo: new	52	\glstreenavigationfmt: new	304
\gls@gobbleopt: new	56	\ifglswrallowprimitivemods: new	176
\glsdoifexistsordo: new	52	list: fixed missing space before	
4.20 (2015-11-30)		description	266
\@gls@link: added		long: fixed typo in \glossentrydesc	270
\@gls@setdefault@glslink@opts	107	super4col: fixed bug in \glossentry	295
added \glsdonohyperlink when		4.23 (2016-04-30)	
hyperlink is suppressed	108	\glscurrentfieldvalue: new	55
\@gls@setdefault@glslink@opts:		\ifglshasfield: added	
new	107	\glscurrentfieldvalue	54, 55
\gls@checkseeallowed@preambleonly:		altlongragged4col: check for	
new	62	nogroupskip changed	284
\glsdonohyperlink: new	117	altsuperragged4col: check for	
4.21 (2016-01-24)		nogroupskip changed	302
\@printglossary: warn if no style has		long: check for nogroupskip changed	270
been set	183	long-booktabs: check for nogroupskip	
General: changed checkfirsthyper		changed	276
assignment	137–143	long3col: check for nogroupskip	
\glossarystyle: set default style if not		changed	271
already set	206	long3col-booktabs: check for	
\glsLTpenaltycheck: new	278	nogroupskip changed	276
\glspatchLToutput: new	279	long4col: check for nogroupskip	
\glspenaltygroupskip: new	279	changed	273
altlong4col-booktabs: new	277	long4col-booktabs: check for	
altlongragged4col-booktabs: new	278	nogroupskip changed	277
long-booktabs: new	275	longragged: check for nogroupskip	
long3col-booktabs: new	276	changed	281
long4col-booktabs: new	276	longragged3col: check for nogroupskip	
longragged-booktabs: new	277	changed	282
longragged3col-booktabs: new	278	super: check for nogroupskip changed	292
\setglossarystyle: set default style if		super3col: check for nogroupskip	
not already set	206	changed	294
4.22 (2016-04-19)		super4col: check for nogroupskip	
\@do@wrglossary: added check for		changed	295
\@arabic	177	superragged: check for nogroupskip	
added test to allow temporary primitive		changed	299
modifications and added arabic case	177	superragged3col: check for	
mcolalttreespannav: new	290	nogroupskip changed	301
mcolindexspannav: new	286	4.24 (2016-05-27)	
mcoltreenonamespannav: new	289	\@gls@extramakeindexopts: new	164
mcoltreespannav: new	288	\@gls@glossary: added check for debug	
\gls@arabicpage: new	176	mode	174
\gls@protected@pagefmts: added		\@gls@see@noindex: new	5
arabic to list	175	debug: new	5
\glsentrytitlecase: new	148	seenoindex: new	6
\glsfindwidesttoplevelname: new	309	\glsnomakeindexwarning: new	40

\GlsSetQuote: new	161	\glstreepredesc: new	304
\GlsSetWriteIstHook: new	161	\glstreesubitem: new	304
4.25 (2016-06-09)		\glstreesubsubitem: new	304
\@gls@enablesavenonumberlist: new	63	4.28 (2017-01-07)	
\@gls@initnonumberlist: new	63	\glspatchtabularx: new	85
\@gls@savenonumberlist: new	63	4.29 (2017-01-19)	
4.26 (2016-10-12)		\@gls@noidx@do: current letter group	
\@glossary@default@style: added		assignment made global	193
check for classicthesis	7	\@print@noidx@glossary: moved	
mcolindex: replaced \@idxitem with		definition of	
\glstreeitem	286	\@gls@currentlettergroup outside	
mcolindexspannav: replaced \@idxitem		of the glossary environment	191
with \glstreeitem	287	General: added check for	
\glstreechildpredesc: new	305	\@glstr@doaccsupp	334
\glstreeitem: new	304	\glsnahyperlinkname: new	261

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in *roman* refer to the code lines where the entry is used.

Symbols	
\!	113, 114
\"	19, 111–114, 116
\#	158
\%	155, 160, 318, 319
\&	31, 152
\'	19
\.	9, 20
\=	19
\?	111–113, 163
\@@delimN	208
\@@do@wrglossary	169, 178
\@@do@wrglossary	153, 175
\@@glo@assign@sortkey	170
\@@glo@list	50
\@@glo@sort	19
\@@glo@type	182
\@@glossarysec	6, 39
\@@glossaryseclabel	7, 39, 195, 196
\@@glossarysecstar	7, 39, 195, 196
\@@gls@checkactual	115
\@@gls@checkbar	114
\@@gls@checkescactual	113
\@@gls@checkescbar	113
\@@gls@checkesclevel	114
\@@gls@checkescquote	112, 163, 164
\@@gls@checklevel	115
\@@gls@checkquote	111, 112, 162
\@@gls@default@entryfmt	94, 103
\@@gls@expand@field	18, 66, 70, 71, 230, 232, 234, 236, 239, 241–243, 365–369
\@@gls@extramakeindexopts	161, 167
\@@gls@fixbraces	180
\@@gls@noexpand@field	18, 65, 66
\@@gls@noidx@no@sanitizesort	19
\@@gls@noidx@nosanitizesort	171
\@@gls@nosanitizesort	18, 171
\@@gls@sanitizesort	18, 171
\@@gls@xdycheckbackslash	116, 117
\@@gls@xdycheckquote	116
\@@gls@localreset	87, 89
\@@gls@localunset	86, 89
\@@gls@reset	87, 89
\@@gls@unset	87, 89
\@@newglossaryentry@defcounters	88
\@@this@glo@	50
\@ACRfull	213
\@ACRfullpl	214
\@ACRlong	141, 213
\@ACRlongpl	143, 214
\@ACRshort	138, 213
\@ACRshortpl	139, 214
\@Acrfull	212
\@Acrfullpl	214
\@Acrlong	141, 213
\@Acrlongpl	143, 214
\@Acrshort	137
\@Acrshortpl	139
\@Alph	175, 177
\@GLS	120
\@GLS@	120, 259
\@GLSdesc	129
\@GLSdesc@	129
\@GLSdescplural	130
\@GLSdescplural@	130
\@GLSfirst	125, 126
\@GLSfirst@	126
\@GLSfirstplural	127
\@GLSfirstplural@	127
\@GLSname	128
\@GLSname@	128
\@GLSpl	122
\@GLSpl@	122, 260
\@GLSplural	126

\@GLSplural@	126	\@Glsuseriii@	133, 134
\@GLSsymbol	130	\@Glsuseriv	134
\@GLSsymbol@	130, 131	\@Glsuseriv@	134
\@GLSsymbolplural	131	\@Glsuserv	135
\@GLSsymbolplural@	131	\@Glsuserv@	135
\@GLStext	124	\@Glsuservi	136
\@GLStext@	124, 125	\@Glsuservi@	136
\@GLSuseri	132	\@Mi	279
\@GLSuseri@	132	\@PGLS	259
\@GLSuserii	133	\@PGLS@	259
\@GLSuserii@	133	\@PGLSpl	259
\@GLSuseriii	134	\@PGLSpl@	259
\@GLSuseriii@	134	\@Pgls	257
\@GLSuseriv	134, 135	\@Pgls@	257
\@GLSuseriv@	135	\@Pglspl	258
\@GLSuserv	135	\@Pglspl@	258
\@GLSuserv@	135	\@Roman	176, 177
\@GLSuservi	136	\@acrfull	212
\@GLSuservi@	136	\@acrfullpl	213
\@Gls	119	\@acrlong	140, 212
\@Gls@	90, 92, 119, 258	\@acrlongpl	142, 213
\@Gls@acrentryname	215	\@acrshort	136, 212, 213
\@Gls@entry@field	72, 145–150	\@acrshortpl	138, 213, 214
\@Gls@entryname	145, 215	\@addtoacronymlists	15
\@Glsdesc	128, 129	\@after	15
\@Glsdesc@	129	\@afterheading	267, 322
\@Glsdescplural	129	\@alph	175, 177
\@Glsdescplural@	129	\@arabic	176, 177
\@Glsfirst	125	\@auxout	56, 57, 91, 167, 169, 172, 181, 185, 261
\@Glsfirst@	125	\@backslashchar	110, 116, 117
\@Glsfirstplural	127	\@before	15
\@Glsfirstplural@	127	\@bsphack	174
\@Glsname	128	\@cGls	91
\@Glsname@	128	\@cGls@	90, 92
\@Glspl	121, 122	\@cGlspl	92
\@Glspl@	90, 93, 122, 258, 259	\@cGlspl@	90, 92
\@Glsplural	126	\@cclv	279
\@Glsplural@	126	\@cgls	91
\@Glsymbol	130	\@cgls@	89, 91
\@Glsymbol@	130	\@cglspl	92
\@Glsymbolplural	131	\@cglspl@	89, 92
\@Glsymbolplural@	131	\@chapter	30
\@Glstext	125	\@classoptionslist	28
\@Glstext@	125	\@closegls	165, 166
\@Glsuseri	132	\@colht	279
\@Glsuseri@	132	\@colroom	279
\@Glsuserii	133	\@currentlabelname	7, 196
\@Glsuserii@	133	\@curroptions	28
\@Glsuseriii	133	\@declaredoptions	28

\@delimN	208	\@glo@desc	60, 76, 77, 79, 81
\@delimR	208	\@glo@descaccess	336–338
\@disable@onlypremakeg	167	\@glo@descplural	60, 76, 77
\@disable@premakecs	30	\@glo@descpluralaccess	336–338
\@disabled@gl saddxdycounters	43	\@glo@do@sortentries	186
\@do@addcounter	41	\@glo@entry	153, 154
\@do@auxoutstuff	184, 185	\@glo@entryprefix	254
\@do@glossentry	201, 334, 335	\@glo@entryprefixfirst	254
\@do@gl s@getcounterprefix	177	\@glo@entryprefixfirstplural ..	254, 255
\@do@gl s@islistofacronyms	15	\@glo@entryprefixplural	254
\@do@gl ssee	82	\@glo@esclabel	84, 85
\@do@ifinlist	41	\@glo@etext	94–96
\@do@newglossaryentry	215, 216,	\@glo@first	61, 77, 80, 81, 241, 369
230, 232, 234, 236–239, 241–246, 365–369		\@glo@firstaccess	335, 337, 338
\@do@seeglossary	169, 180	\@glo@firstplural	61, 77, 80, 81, 369
\@do@subglossentry	203, 335	\@glo@firstpluralaccess	336–338
\@do@wrglossary	108	\@glo@grabfirst	192
\@do@writeaux@info	181	\@glo@label	
\@ehc	279	64, 71–82, 88, 152, 254, 255, 309, 310, 338	
\@empty ... 12, 13, 15, 27, 28, 30, 41, 42, 46,		\@glo@list	82
48, 49, 79, 84, 108, 109, 119–123, 137–		\@glo@long	53, 64, 78, 81
143, 156, 159, 161, 165, 166, 173, 174,		\@glo@longaccess	336–338
179, 196, 198, 206, 231, 233, 235, 237–		\@glo@longpl	64,
240, 242, 244, 246, 316, 318, 320, 352–354		78, 81, 230, 232, 234, 236, 238, 241, 243, 365	
\@end@fixbraces	180	\@glo@longpluralaccess	336–338
\@endfortrue	24, 52, 70, 262	\@glo@name	11, 60, 65, 77, 80, 81
\@esphack	175	\@glo@no@assign@sortkey	168
\@expandtwoargs	28	\@glo@nonumberlist	63
\@firstofone	19, 20	\@glo@numfmt	178, 316
\@firstofthree	105, 118,	\@glo@parent	12, 62, 78–80, 84, 85, 187
119, 121, 123, 137, 138, 140, 142, 352–354		\@glo@plural	61, 77, 80, 367
\@firstoftwo	22,	\@glo@pluralaccess	336–338
23, 69, 70, 105, 121–123, 138–140, 142, 143		\@glo@prefix	
\@for	23, 28,	8, 62, 78, 84, 85, 109, 110, 178, 315, 316	
30, 41, 43, 49, 50, 68, 70, 110, 156, 158,		\@glo@range	178, 315, 316
167, 168, 175, 180, 186, 217, 230, 233,		\@glo@see	62, 78, 82
235, 237, 239, 242, 244, 246, 262, 263, 316		\@glo@seeautonumberlist	8, 62
\@glo@@desc	81	\@glo@short	54, 64, 78, 81, 368
\@glo@@symbol	82	\@glo@shortaccess	336–338, 365–368
\@glo@access	335, 337, 338, 340	\@glo@shortpl	64, 78, 81,
\@glo@addchildren	186, 190	230, 232, 234, 236, 238, 241, 243, 365, 368	
\@glo@assign@sortkey	168, 170, 197	\@glo@shortpluralaccess	336–338
\@glo@check@mkidxrangechar		\@glo@sort	11, 19, 60, 78, 80, 84, 85
109, 110, 178, 315, 316		\@glo@sortedinsert	187
\@glo@childlist	186	\@glo@sortentries	188–190
\@glo@counter	62, 78, 81	\@glo@sorthandler@case	189
\@glo@counterprefix 172, 177–179, 206, 209		\@glo@sorthandler@letter	189
\@glo@default@sorttype .. 10, 170, 188–190		\@glo@sorthandler@nocase	190
\@glo@defaultcounter	81	\@glo@sorthandler@word	188

\@glo@sortinghandler	186, 187	\@gls@checkactual	111, 163
\@glo@sortinglist	186, 187, 190	\@gls@checkbar	111, 163
\@glo@sorttype	170, 191, 192, 198	\@gls@checkedmkidx	110–116, 162–164
\@glo@storeentry	11, 13	\@gls@checkescactual	111, 163
\@glo@sufffix	109, 110, 178, 316	\@gls@checkescbar	111, 163
\@glo@symbol		\@gls@checkescquote	111, 162–164
.....	53, 61, 77, 82, 235, 236, 240, 241, 337	\@gls@checklevel	111, 163
\@glo@symbolaccess	336–338, 368	\@gls@checkmkidxchars	
\@glo@symbolplural	61, 77, 82	84, 109, 162, 169, 177, 179, 315, 316
\@glo@symbolpluralaccess	336–338	\@gls@checkquote	111, 162
\@glo@text	60,	\@gls@classI	156, 157
77, 80, 82, 119–123, 144–146, 236, 255, 367		\@gls@classII	156, 157
\@glo@textaccess ...	335, 337, 338, 365–368	\@gls@codepage	185
\@glo@thislabel	83	\@gls@counter	
\@glo@thislettergrp	191–193	104, 107–109, 153, 172, 178, 179, 316
\@glo@thisvalue	54, 55	\@gls@counterwithin	10, 196, 198
\@glo@tmp	71, 72, 179	\@gls@ctr	41
\@glo@type	7,	\@gls@currentlettergroup	191, 193
12, 13, 61, 77–79, 81, 82, 153, 167, 173,		\@gls@declareoption	
182–187, 190, 191, 195, 196, 215, 231,		8, 9, 13, 14, 17, 22, 25–28
233, 235, 237, 239, 242, 244, 246, 261–263		\@gls@default	93
\@glo@types	49,	\@gls@default@value	
50, 57, 87, 88, 153, 154, 167, 168, 251, 309		53–55, 65, 66, 77, 78, 80, 81, 240, 254
\@glo@useri	63, 78, 81	\@gls@deffile	68, 69
\@glo@useriii	63, 78, 81	\@gls@defsort	11, 12, 82
\@glo@useriii	63, 78, 81	\@gls@defsortcount	11, 12, 58
\@glo@useriv	64, 78, 81	\@gls@do@acronymsdef	14, 29, 59
\@glo@userv	64, 78, 81	\@gls@do@indexdef	28, 29, 59
\@glo@uservi	64, 78, 81	\@gls@do@numbersdef	28, 29, 59
\@glodesc	81	\@gls@do@symbolsdef	27, 59
\@glolist@	79	\@gls@do@symbolssdef	29
\@gloname	81	\@gls@doautomake	26, 168
\@glossary@default@style	7, 9, 183, 206, 247	\@gls@docheckquotedef	162–164
\@glossaryentryfield	84	\@gls@docloadedfalse	4
\@glossarysection	37	\@gls@docloadedtrue	4
\@glossarystyle	183, 195	\@gls@dodeflistparser	168
\@glossarysubentryfield	84, 85	\@gls@doentrydef	103
\@gls	118	\@gls@dolast	180, 181
\@gls@	89, 91, 118, 257, 258	\@gls@donext	180, 181
\@gls@@link	106	\@gls@donext@def	152
\@gls@Hcounter	108, 109	\@gls@dothiswrite	165, 166
\@gls@ReturnAfterFi	209	\@gls@elem	262
\@gls@access@display	340, 341	\@gls@enablesavenonumberlist	68
\@gls@actualchar ..	84, 85, 113, 115, 160, 319	\@gls@encapchar	
\@gls@addpredefinedattributes	155, 164	113, 114, 160, 178, 179, 316, 319
\@gls@adjustmode	153	\@gls@entry@count	90, 91
\@gls@automake	168	\@gls@entry@field	
\@gls@between	262, 263	71, 72, 89, 145–151, 338–340
\@gls@body	145	\@gls@escbsdq	111, 161, 320

<code>\@gls@expand@fields</code>	66, 67	<code>\@gls@noidx@do</code>	191
<code>\@gls@expandonce</code>	67	<code>\@gls@noidx@getgrouptitle</code>	169, 205
<code>\@gls@extramakeindexopts</code>	167	<code>\@gls@noidx@sanitizesort</code>	19, 171
<code>\@gls@fetchfield</code>	55	<code>\@gls@noidx@setsanitizesort</code>	21, 171
<code>\@gls@field@link</code>	72, 73, 124–136	<code>\@gls@noidx@loclist@finalsep</code>	171
<code>\@gls@firsttok</code>	191, 192	<code>\@gls@noidx@loclist@prev</code>	171, 193, 194
<code>\@gls@fixbraces</code>	82	<code>\@gls@noidx@loclist@sep</code>	170, 193, 194
<code>\@gls@forbidtexext</code>	57	<code>\@gls@noref@warn</code>	169, 191
<code>\@gls@get@counterprefix</code>	179	<code>\@gls@numberlink</code>	208, 209
<code>\@gls@getbody</code>	145	<code>\@gls@numbersdef</code>	28
<code>\@gls@getcounterprefix</code>	177	<code>\@gls@numlist@lastsep</code>	152
<code>\@gls@getgrouptitle</code>	169, 204, 263	<code>\@gls@numlist@nextsep</code>	152
<code>\@gls@glossary</code>	174	<code>\@gls@numlist@sep</code>	152
<code>\@gls@gobbleopt</code>	56	<code>\@gls@old@chapter</code>	30
<code>\@gls@grptitle</code>	204, 261, 263	<code>\@gls@oldnewglossaryentryposthook</code> .	337
<code>\@gls@hyp@opt</code>	72, 73, 91, 92, 106, 118–143, 212–214, 256–259	<code>\@gls@oldnewglossaryentryprehook</code> ..	337
<code>\@gls@hyp@opt@cs</code>	105	<code>\@gls@onlypremakeg</code>	30
<code>\@gls@hypergroup</code>	261	<code>\@gls@order</code>	165, 166
<code>\@gls@ifinlist</code>	41	<code>\@gls@org@LT@output</code>	279
<code>\@gls@ifnotmeasuring</code>	85	<code>\@gls@org@glsnoidxdisplayloc</code>	171
<code>\@gls@igtype</code>	60	<code>\@gls@org@glsseeformat</code>	171
<code>\@gls@increment@currcount</code>	89	<code>\@gls@patchtabularx</code>	85
<code>\@gls@indexdef</code>	28	<code>\@gls@preglossaryhook</code>	184
<code>\@gls@initnonumberlist</code>	63, 78	<code>\@gls@prevlevel</code> .	289–291, 310–313, 328, 329
<code>\@gls@islistofacronyms</code>	15	<code>\@gls@provide@newglossary</code>	57
<code>\@gls@keylist</code>	364	<code>\@gls@quotechar</code>	112–115, 160–164, 319
<code>\@gls@keymap</code>	63, 68, 70, 71, 254, 337	<code>\@gls@reference</code>	169, 172
<code>\@gls@label</code>	169, 172, 178	<code>\@gls@removespaces</code>	209
<code>\@gls@langmod</code>	165, 166	<code>\@gls@renewglossary</code>	165
<code>\@gls@levelchar</code>	85, 114, 115, 160, 319	<code>\@gls@replacementtext</code>	340
<code>\@gls@link</code> ..	106, 119–124, 137–144, 352–354	<code>\@gls@rest</code>	145
<code>\@gls@link@checkfirsthyper</code>	118–123	<code>\@gls@roman</code>	44, 45, 316, 317
<code>\@gls@link@label</code>	107, 232, 238	<code>\@gls@sanitized@tmp</code>	110
<code>\@gls@link@nocheckfirsthyper</code> ..	124, 137–143	<code>\@gls@sanitizedesc</code>	24
<code>\@gls@link@opts</code>	107, 232, 238	<code>\@gls@sanitizesort</code>	11
<code>\@gls@list</code>	262, 263	<code>\@gls@sanitizesymbol</code>	24, 25
<code>\@gls@listsuffix</code>	41	<code>\@gls@saveentrycounter</code>	108, 153
<code>\@gls@loadlist</code>	9, 246	<code>\@gls@savenonumberlist</code>	62, 63
<code>\@gls@loadlong</code>	8, 9, 246	<code>\@gls@see@noindex</code>	6, 62
<code>\@gls@loadsuper</code>	9, 246	<code>\@gls@setacrstyle</code>	24, 25, 29
<code>\@gls@loadtree</code>	9, 247	<code>\@gls@setcounter</code>	58
<code>\@gls@local@increment@currcount</code>	89	<code>\@gls@setdefault@glslink@opts</code>	107
<code>\@gls@loclist</code>	170, 171, 192, 193	<code>\@gls@setsort</code>	11, 12, 108
<code>\@gls@map</code>	68–70	<code>\@gls@setupshortcuts</code>	29
<code>\@gls@missingnumberlist</code>	81	<code>\@gls@sort</code>	193
<code>\@gls@noaccess</code>	340	<code>\@gls@sort@A</code>	187, 188
<code>\@gls@noexpand@fields</code>	67	<code>\@gls@sort@B</code>	187, 188
<code>\@gls@nohyperlist</code>	16, 59, 107	<code>\@gls@startswithexpandonce</code>	66
		<code>\@gls@storenonumberlist</code>	63, 81

\@gls@symbolsdef	27	\@glslink	108, 118, 152, 261
\@gls@this	175	\@glslocalreset	86, 89
\@gls@thisHloc	179	\@glslocalunset	86, 89
\@gls@thisfield	55	\@glslocref	172, 177, 178, 315, 316
\@gls@thislabel	52, 180, 190	\@glsminrange	155–157, 317
\@gls@thislist	152	\@glsname	127
\@gls@thisloc	179	\@glsname@	127, 128
\@gls@thisval	70	\@glsnextpages	183
\@gls@title	37	\@glsnodelc	77, 79, 81
\@gls@tmp . 12, 13, 32, 46, 67, 110, 174, 262, 263		\@glsnoname	77, 80, 81
\@gls@tmpb	111–116, 162–164	\@glsnonextpages	183
\@gls@toc	39	\@glsnumberformat	
\@gls@type	168, 217, 230, 233, 235, 237, 239, 242, 244, 246, 309	104, 107, 153, 172, 178, 315, 316
\@gls@updatechecked	110, 111, 162, 163	\@glsopenfile	165, 173
\@gls@usetranslator	22, 23, 32	\@glsorder	167
\@gls@value	65, 66, 148	\@glspl	121
\@gls@warnonglossdefined	17, 182	\@glspl@	90, 92, 121, 257–259
\@gls@warnonthehglossdefined	17, 201	\@glsplural	126
\@gls@write@entrycounts	90	\@glsplural@	126
\@gls@writedef	68	\@glsreset	86, 89
\@gls@writeisthook	159, 161	\@glssee	82, 180
\@gls@xdy@locationlist	156	\@glsymbol	130
\@gls@xdycheckbackslash	110	\@glsymbol@	130
\@gls@xdycheckquote	110	\@glsymbolplural	131
\@gls@xref	179	\@glsymbolplural@	131
\@gls@Alphacompositor	35, 45, 317	\@glstarget	118, 201, 261
\@gls@Hlocref	177	\@glstext	124
\@gls@acronymlists 15, 16, 49, 215, 217, 230, 231, 233, 235, 237, 239, 242, 244, 246, 251		\@glstext@	124
\@gls@addkey	71	\@glsunset	86, 89
\@gls@addstoragekey	70	\@glsuseri	132
\@gls@addxdyattribute	42, 43	\@glsuseri@	132
\@gls@defaultsort	11	\@glsuserii	132
\@gls@desc	128	\@glsuseriii	133
\@gls@desc@	128	\@glsuseriii@	133
\@gls@descplural	129	\@glsuseriv	134
\@gls@descplural@	129	\@glsuseriv@	134
\@gls@disp	123	\@glsuserv	135
\@gls@entry	87, 88, 91	\@glsuserv@	135
\@gls@entrytitlecase	148	\@glsuservi	136
\@gls@first	125	\@glsuservi@	136
\@gls@first@	125	\@glswidestname	310, 311, 328
\@gls@firstletter	49, 155	\@glswritefiles	27
\@gls@firstplural	127	\@glsxtr@doaccsupp	334
\@gls@firstplural@	127	\@gobble	12, 68, 69, 85, 110, 154, 155, 158, 169, 314, 318, 319
\@gls@hypernumber	208	\@idxitem	304
\@gls@isacronymlistfalse	16	\@ifclassloaded	4, 10, 38
\@gls@isacronymlisttrue	16	\@ifnextchar	58, 105

<code>\@ifpackageloaded</code>	<code>\@printglossary</code>	182
..... 4, 7, 22, 23, 32, 48, 85, 151, 161, 334	<code>\@roman</code>	44, 316
<code>\@ifstar</code>	<code>\@secondofthree</code>	
<code>\@ifundefined</code>	105, 118, 119, 122, 137, 139, 141, 143, 352	
. 32, 262, 269, 280, 291, 298, 311, 328, 342	<code>\@secondoftwo</code>	22, 23, 32, 68, 70, 117–
<code>\@ignored@glossaries</code>	120, 123, 137, 138, 140, 141, 352–354, 372	
59, 60	<code>\@set@glo@numformat</code>	178, 316
<code>\@input@</code>	<code>\@sglsaddkey</code>	71
184	<code>\@sglsaddstoragekey</code>	70
<code>\@istfilename</code>	<code>\@thirdofthree</code>	
167 105, 120, 123, 138, 140, 141, 143, 352	
<code>\@makecol</code>	<code>\@this@attr</code>	158
279	<code>\@this@childlabel</code>	186
<code>\@makeglossary</code>	<code>\@this@counter</code>	43
167	<code>\@this@ctr</code>	158
<code>\@minus</code>	<code>\@this@key</code>	70
266, 285, 304	<code>\@this@label</code>	186
<code>\@mkboth</code>	<code>\@thiscs</code>	30
38	<code>\@tmp</code>	44, 317
<code>\@newglossary</code>	<code>\@use@option</code>	28
56, 57	<code>\@warn@nomakeglossaries</code>	166, 185
<code>\@newglossaryentry@defcounters</code> ..	<code>\@wrglossary@pageformat</code>	176
82, 88	<code>\@wrglossarynumberhook</code>	176, 177
<code>\@newglossaryentryposthook</code>	<code>\@xdy@main@language</code>	25, 165, 185
..... 71, 72, 83, 254, 337	<code>\@xdy@attributelist</code>	42, 158
<code>\@newglossaryentryprehook</code>	<code>\@xdy@attributes</code>	42, 156, 314, 316
..... 71, 76, 78, 254, 337	<code>\@xdy@counters</code>	41, 43, 158
<code>\@nil</code>	<code>\@xdy@language</code>	185
15, 82, 109–111, 145, 162,	<code>\@xdy@lettergroups</code>	49, 159, 319
163, 178, 180, 191–193, 208, 209, 315, 316	<code>\@xdy@locationclassorder</code>	47, 157, 318
<code>\@nnil</code>	<code>\@xdy@locref</code>	42, 159, 314, 318
15, 180	<code>\@xdy@requiredstyles</code>	47, 156, 316
<code>\@no@makeglossaries</code>	<code>\@xdysortrules</code>	47, 159, 319
168, 170	<code>\@xdystyle</code>	156, 316
<code>\@no@post@desc</code>	<code>\@xdyuseralphabets</code>	44, 156, 316
321	<code>\@xdyuserlocationdefs</code> ...	46, 157, 315, 317
<code>\@nopostdesc</code>	<code>\@xdyuserlocationnames</code>	46, 315
183	<code>\@xfor@nextelement</code>	180
<code>\@onelevel@sanitize</code>	<code>\</code>	83, 110, 154, 160,
19,	161, 208, 209, 319, 320, 322–324, 332, 333	
44, 68, 84, 110, 159, 179, 181, 192, 317, 318	<code>\{</code>	68, 69, 154, 160, 161, 314, 319, 320
<code>\@onlypreamble</code> ...	<code>\}</code>	69, 154, 160, 161, 314, 320
58, 68, 77, 90, 93, 168, 171	<code>\^</code>	19
<code>\@onlypremakeg</code>	<code>\‘</code>	19
34–36, 42, 43, 46, 58, 161	<code>\ </code>	111, 113, 163
<code>\@org@glossaryentrynumbers</code>	<code>\~</code>	20
183, 184		
<code>\@org@gls@assign@descplural</code>		
..... 230, 239, 241–244, 365, 368, 369		
<code>\@org@gls@assign@firstpl</code>		
230,		
232–234, 236, 237, 239, 241–244, 365–369		
<code>\@org@gls@assign@plural</code>		
230, 232, 234, 236–239, 241–244, 365–369		
<code>\@org@gls@assign@symbolplural</code> ..		
230,		
232–234, 236, 237, 241–244, 366, 367, 369		
<code>\@org@glsnumberformat</code>		
152		
<code>\@org@newglossaryentryprehook</code>		
76		
<code>\@outputpage</code>		
279		
<code>\@p@glossarysection</code>		
37		
<code>\@pgls</code>		
256		
<code>\@pgls@</code>		
256		
<code>\@pglspl</code>		
257		
<code>\@pglspl@</code>		
257		
<code>\@plus</code>		
266, 285, 304		
<code>\@print@glossary</code>		
182		
<code>\@print@noidx@glossary</code>		
182		
<code>\@printgloss@setsort</code>		
168, 170, 183		

A

<code>\AA</code>	20
------------------------	----

<code>\aa</code>	20	<code>\andname</code>	181
<code>accsupp package</code>	334	<code>\AnyTrackedLanguages</code>	33, 372
<code>\accsuppglossaryentryfield</code>	334	<code>\appto</code>	16, 63, 70–72, 254, 337
<code>\accsuppglossarysubentryfield</code>	335	<code>array package</code>	275, 280, 298
<code>\acrfootnote</code>	232, 238	<code>article class</code>	178
<code>\Acrfull</code>	229	<code>\AtBeginDocument</code>	14, 48, 68, 85, 153, 169
<code>\acrfull</code>	229	<code>\AtEndDocument</code>	26, 68, 90, 169, 173, 184, 185, 262
<code>\ACRfullfmt</code>	213, 216, 224, 226, 360, 362	B	
<code>\Acrfullfmt</code>	213, 216, 224, 226, 360, 362	<code>\b</code>	20
<code>\acrfullfmt</code>	212, 216, 224, 226, 360, 362	<code>babel package</code>	22, 30, 32, 48
<code>\acrfullformat</code>	151, 212, 230, 245	<code>\begin</code>	158, 191, 266, 269–272, 274, 275, 277, 278, 280–303, 318
<code>\Acrfullpl</code>	229	<code>\BeginAccSupp</code>	340
<code>\acrfullpl</code>	229	<code>\begingroup</code>	5, 174, 176, 209
<code>\ACRfullplfmt</code> ...	214, 216, 224, 226, 360, 362	<code>\bfseries</code>	270– 274, 276, 277, 281–285, 293–297, 299–303
<code>\Acrfullplfmt</code> ...	214, 216, 224, 226, 360, 362	<code>\bgroup</code>	19, 76, 151, 183, 186
<code>\acrfullplfmt</code> ...	213, 216, 224, 226, 360, 362	<code>booktabs package</code>	275–278
<code>\acrlinkfootnote</code>	231	<code>\boolean</code>	244
<code>\acrlinkfullformat</code>	212–214	<code>\boolfalse</code>	27
<code>\Acrlong</code>	228	<code>\booltrue</code>	27
<code>\acrlong</code>	228	<code>\bottomrule</code>	276, 277
<code>\Acrlongpl</code>	228	<code>\box</code>	279
<code>\acrlongpl</code>	228	C	
<code>\acrnameformat</code>	236, 367	<code>\c</code>	20
<code>\acronymentry</code>	215, 218–223, 225–228, 356–358, 361–364	<code>\c@equation</code>	108
<code>\acronymfont</code>	100, 101, 137–140, 145, 151, 214–216, 218–228, 231–233, 235, 237–242, 349, 350, 352–354, 356–358, 360–364, 366–368	<code>\c@glossarysubentry</code>	197
<code>\acronymname</code>	14, 33	<code>\c@page</code>	175–177
<code>\acronymsort</code>	215, 218–223, 225–228, 356–358, 361, 362, 364	<code>\cGls</code>	92
<code>\acronymtype</code>	14, 215, 217, 230–239, 241–246, 365–368	<code>\cgls</code>	91
<code>\acrpluralsuffix</code>	216, 218– 221, 225–227, 230–234, 236–239, 241– 243, 245, 246, 356, 357, 361–363, 365–369	<code>\cGlsformat</code>	90
<code>\Acrshort</code>	228	<code>\cglsformat</code>	89
<code>\acrshort</code>	228	<code>\cGlspl</code>	93
<code>\Acrshortpl</code>	228	<code>\cglspl</code>	92
<code>\acrshortpl</code>	228	<code>\cGlsplformat</code>	90
<code>\addcontentsline</code>	40	<code>\cglsplformat</code>	89
<code>\addglossarytocaptions</code>	32	<code>\char</code>	205
<code>\addtolength</code>	311, 328	<code>classicthesis package</code>	7
<code>\advance</code>	12, 13, 79, 108, 279	<code>\cleardoublepage</code>	40
<code>\AE</code>	20	<code>\clearpage</code>	39, 40
<code>\ae</code>	20	<code>\closeout</code>	68, 159, 161, 165, 173
<code>amsgen package</code>	4, 104	<code>\compatglossarystyle</code>	321–333
<code>amsmath package</code>	85	<code>\compatibleglossentry</code>	203
		<code>\compatiblesubglossentry</code>	203
		<code>\copy</code>	279
		<code>\count@</code>	192

<code>\csdef</code>	18, 70–73, 82, 83, 88, 89, 186, 187, 207, 217, 320
<code>\csedef</code>	90, 176
<code>\csgdef</code>	37, 56, 59, 89, 90, 181, 194, 195
<code>\cslet</code>	63, 76, 77, 83, 190
<code>\csname</code>	10–13, 28, 31–33, 39, 42, 44, 45, 48, 50, 52, 57–59, 65, 66, 70–74, 76, 79–85, 87, 103, 107–109, 119–123, 137–144, 152, 153, 156–158, 162–165, 169, 172–174, 176, 178, 179, 182–185, 187, 195, 201, 203, 206, 207, 210, 247– 255, 262, 263, 310, 311, 314–316, 328, 334, 335, 338, 339, 342, 352–354, 370, 371
<code>\csshow</code>	251
<code>\csuse</code>	33, 37, 56, 65, 66, 72, 73, 103, 166, 187, 189, 191, 192, 194, 196, 197, 206, 218, 255, 321–333
<code>\csxdef</code>	81, 90
<code>\currentglossary</code>	37, 183, 196, 198
<code>\currentglssubentry</code>	197, 199
<code>\CurrentOption</code>	28, 254, 334
<code>\CurrentTrackedLanguage</code>	33, 372, 373
<code>\CurrentTrackedTag</code>	33, 372, 373
<code>\CustomAcronymFields</code>	246
<code>\CustomNewAcronymDef</code>	246
D	
<code>\d</code>	19
<code>datatool package</code>	187
<code>\day</code>	156, 160, 316, 319
<code>\DeclareAcronymList</code>	14, 16, 215, 217, 231, 233, 235, 237, 239, 242, 244, 246
<code>\DeclareListParser</code>	168
<code>\DeclareOption</code>	8, 254, 334
<code>\DeclareOptionX</code>	8
<code>\DeclareRobustCommand</code>	34, 180, 181, 240, 340–342
<code>\def</code>	8, 11, 12, 15, 19, 20, 25, 26, 29–31, 33, 34, 37, 41, 44–49, 52, 56–58, 60–64, 67, 74, 76, 78–83, 85, 89–93, 103, 104, 107–144, 152, 153, 155, 160–167, 169–173, 176–180, 182, 183, 186, 190–199, 205–210, 212–215, 230– 237, 239–244, 246, 254, 256–260, 262– 265, 289–291, 310–313, 315, 316, 319, 321, 328, 329, 334–337, 351–354, 365–369
<code>\def@glx@xdycheckbackslash</code>	116, 117
<code>\DefaultNewAcronymDef</code>	231
<code>\defglstryfmt</code>	58, 59, 103, 217, 229, 231, 234, 235, 237, 240, 243, 245
<code>\define@boolkey</code>	5, 6, 8–10, 14, 20, 21, 24–27, 104, 197
<code>\define@choicekey</code>	6, 7, 10, 21, 23, 25, 62, 195–197
<code>\define@key</code>	7, 10, 16, 21, 25, 26, 60– 64, 70, 71, 104, 153, 195, 197, 254, 335, 336
<code>\DefineAcronymSynonyms</code>	29, 229
<code>\delimN</code>	158, 168, 194, 208, 209, 318
<code>\delimR</code>	158, 208, 318
<code>\DescriptionDUANewAcronymDef</code>	235
<code>\DescriptionFootnoteNewAcronymDef</code>	233
<code>\descriptionname</code>	33, 270– 274, 276, 277, 281–285, 293–297, 299–303
<code>\DescriptionNewAcronymDef</code>	237
<code>\dimen@</code>	219, 279, 309
<code>\disable@keys</code>	29
<code>\do</code>	23, 28, 30, 41, 43, 49, 50, 68, 70, 110, 152, 156, 158, 167, 168, 175, 180, 186, 217, 230, 233, 235, 237, 239, 242, 244, 246, 262, 263, 316
<code>\do@gl@storeentry</code>	11, 12, 82
<code>\do@gl@link@checkfirsthyper</code>	106, 107, 118–124, 137–143, 352, 353
<code>\do@gl@xdycheckbackslash</code>	110
<code>\do@gl@disablehyperinlist</code>	107
<code>\do@gl@shaschildren</code>	52
<code>doc package</code>	4, 5, 13
<code>\doifglossarynoexistsordo</code>	57
<code>\dtl@ifsingle</code>	205
<code>\dtl@insertinto</code>	187
<code>\dtl@sortresult</code>	187, 188
<code>\dtlcompare</code>	188
<code>\dtlicompare</code>	188
<code>\DTLifinlist</code>	60, 107
<code>\DTLifint</code>	205
<code>\dtlletterindexcompare</code>	188
<code>\DTLsubstituteall</code>	110
<code>\dtlwordindexcompare</code>	187
<code>\DUANewAcronymDef</code>	244
E	
<code>\eappto</code>	59, 83, 176
<code>\edef</code>	12, 15, 30, 33, 41, 42, 44–47, 50, 52, 57, 59, 60, 65, 66, 70, 73–78, 83, 84, 103, 107– 116, 152–155, 160–166, 168, 169, 173, 177–179, 181, 184–188, 192, 197, 199, 205, 209, 210, 230, 232, 234, 236, 238, 241, 243, 261, 314, 315, 317, 319, 364–368
<code>\egroup</code>	19, 77, 152, 184, 186

else	5, 9, 12–15, 17, 18, 20, 21, 26–30, 34, 35, 38, 40–44, 46–49, 62, 65, 79, 80, 83–85, 89, 90, 106–108, 110–117, 119–124, 145, 155, 156, 159, 161–166, 173–180, 183, 192, 196–200, 206, 208, 209, 219, 233, 235, 237, 239, 240, 242–244, 247, 262, 266, 270, 271, 273, 276, 277, 279, 281, 282, 284, 292, 294, 295, 299, 301, 302, 305–307, 309–312, 315–321, 326–329, 340	38, 40–49, 58, 62, 65, 79–85, 89, 90, 106–117, 119–124, 146, 153, 155, 156, 159, 161–166, 168, 173–181, 183, 185, 192, 196–200, 206–209, 219, 229, 231, 233, 235, 237, 239, 240, 242–247, 253, 262, 266, 270, 271, 273, 276–279, 281, 282, 284, 292, 294, 295, 299, 301, 302, 306–312, 314–318, 320, 321, 326–329, 334, 340	
\emph	180, 210	file types	
\empty	209, 334	.aux	184
\end	158, 191, 266, 269–272, 274, 275, 277, 278, 280–304, 318	.glo	84
\end@doifinlist	41	.ist	154, 164
\end@getprefix	179	.toc	40
\end@glscislistofacronyms	15	.xdy	35
\EndAccSupp	340	glo	252
\endcsname	10–13, 28, 31–33, 39, 42, 44, 45, 48, 50, 52, 57–59, 65, 66, 70–74, 76, 79–85, 87, 103, 107–109, 119–123, 137–144, 152, 153, 156–158, 162–165, 169, 172–174, 176, 178, 179, 182–185, 187, 195, 201, 203, 206, 207, 210, 247–255, 262, 263, 310, 311, 314–316, 328, 334, 335, 338, 339, 342, 352–354, 370, 371	\firstacronymfont	102, 218–220, 225, 226, 231, 236, 238, 241, 351, 355–357, 361, 362
\endfoot	270–272, 274, 276, 277, 281–283, 285	\footnote	225, 226, 231, 362
\endgroup	5, 175, 177, 209	\FootnoteNewAcronymDef	239
\endhead	270–274, 276, 277, 281–285	\forallallglossaries	50, 173, 182, 309
\endtheglossary	5	\forallallglstentries	87, 88, 91, 153, 154
\entryname	33, 270–274, 276, 277, 281–285, 293–297, 299–303	\ForEachTrackedDialect	33, 372, 373
\equal	21, 29, 39, 108, 167, 205, 262	\forglstentries	50, 52, 83, 190, 309
equation (counter)	108, 109	\forlistcsloop	186, 191
etoolbox package	4	\forlistloop	171, 193
\expandafter	11–13, 19, 28, 30, 32, 33, 42, 44, 45, 47–50, 52, 57, 58, 60, 65, 66, 68–74, 76, 79, 80, 82, 84, 85, 87, 103, 107–116, 145, 152, 154, 155, 158, 162, 163, 165, 173–175, 177, 178, 180, 183, 187, 192, 193, 201–203, 207, 209, 210, 232, 238, 247–252, 254, 255, 262, 263, 310, 314–316, 318, 319, 334, 335, 338, 340, 364, 370, 371	G	
\expandonce	65–67, 110, 162–164, 176, 188, 201, 203, 215, 230, 232, 234, 236, 238, 241, 243, 334, 335	garamondx package	211
		\gdef	12, 42, 57, 74, 79, 80, 174, 198, 262
		\Genacrffullformat	101, 216, 218, 219, 225, 351, 355, 356, 362
		\genacrffullformat	101, 102, 216, 218, 219, 225, 350, 351, 355, 356, 361
		\GenericAcronymFields	216, 218, 219, 221–225, 227, 228, 355–358, 360, 361, 364
		\Genplacrffullformat	101, 216, 218–220, 225, 350, 356, 362
		\genplacrffullformat	101, 102, 216, 218, 219, 225, 350, 356, 362
		\glo@desc	321
		\glo@do@compare	187, 188
		\glo@grabfirst	193
		\glo@label	52, 83
		\glo@list	83
		\glo@name	202
		\glo@parent	52
		\glo@type	83
		\glo@value	68, 69
		\global	12, 13, 65, 68, 76, 77, 82, 87, 174, 184, 192, 193, 198, 279
\fi	5–7, 9, 11–15, 17, 18, 20, 21, 23, 26–30, 34, 35,		

<code>\glolinkprefix</code>	108, 152, 201	<code>long-booktabs</code>	275, 278
<code>glossareentry (counter)</code>	199	<code>long3col</code>	271, 272, 276, 323
<code>glossaries package</code>		<code>long3col-booktabs</code>	276, 278
.....	28, 48, 155, 246, 254, 266, 314, 334	<code>long3colborder</code>	272, 323
<code>glossaries-accsupp package</code>	83, 334	<code>long3colheader</code>	272, 276, 323
<code>glossaries-extra package</code>	334	<code>long3colheaderborder</code>	272, 323
<code>\GlossariesWarning</code> 5, 6, 17, 20, 21, 37, 40,		<code>long4col</code>	272–274, 277, 323
51, 55, 62, 65, 91–93, 103, 105, 151, 166,		<code>long4col-booktabs</code>	276, 277
169–172, 175, 179, 183, 203, 206, 314, 334		<code>long4colborder</code>	273, 324
<code>\GlossariesWarningNoLine</code>		<code>long4colheader</code>	273, 276, 324
.....	5, 17, 167–169, 173, 185, 262	<code>long4colheaderborder</code>	274, 324
<code>\glossary</code>	315, 316	<code>longborder</code>	270, 323
<code>glossary package</code>	1, 210	<code>longheader</code>	270, 275, 276, 323
<code>glossary styles:</code>		<code>longheaderborder</code>	271, 323
<code>altlist</code>	267, 268, 322	<code>longragged</code>	277, 280–282
<code>altlistgroup</code>	268, 322	<code>longragged-booktabs</code>	277
<code>altlisthypergroup</code>	268, 322	<code>longragged3col</code>	278, 282, 283, 325
<code>altlong4col</code>	274, 275, 283, 324	<code>longragged3col-booktabs</code>	278
<code>altlong4col-booktabs</code>	277, 278	<code>longragged3colborder</code>	282, 325
<code>altlong4colborder</code>	275, 324	<code>longragged3colheader</code>	283, 325
<code>altlong4colheader</code>	274, 277, 324	<code>longragged3colheaderborder</code>	283, 325
<code>altlong4colheaderborder</code>	275, 324	<code>longraggedborder</code>	281, 324
<code>altlongragged4col</code> ...	278, 283, 284, 325	<code>longraggedheader</code>	281, 325
<code>altlongragged4col-booktabs</code>	278	<code>longraggedheaderborder</code>	281, 325
<code>altlongragged4colborder</code>	284, 325	<code>mcolalttree</code>	290, 330
<code>altlongragged4colheader</code>	284, 325	<code>mcolalttreegroup</code>	290, 330
<code>altlongragged4colheaderborder</code>	285, 326	<code>mcolalttreehypergroup</code>	290, 330
<code>altsuper4col</code>	296, 297, 302, 333	<code>mcolindex</code>	286, 329
<code>altsuper4colborder</code>	297, 333	<code>mcolindexgroup</code>	286, 329
<code>altsuper4colheader</code>	297, 333	<code>mcolindexhypergroup</code>	286, 329
<code>altsuper4colheaderborder</code> ...	297, 333	<code>mcoltree</code>	287, 329
<code>altsuperragged4col</code>	302, 303, 331	<code>mcoltreegroup</code>	329
<code>altsuperragged4colborder</code> ...	303, 331	<code>mcoltreehypergroup</code>	287, 288, 329
<code>altsuperragged4colheader</code> ...	303, 331	<code>mcoltreenoname</code>	288, 330
<code>altsuperragged4colheaderborder</code> .		<code>mcoltreenonamegroup</code>	289, 330
.....	303, 331	<code>mcoltreenonamehypergroup</code> ...	289, 330
<code>alttree</code>	289, 304, 305, 310, 312, 328	<code>sublistdotted</code>	322
<code>alttreegroup</code>	312, 329	<code>super</code>	291–293, 299, 332
<code>alttreehypergroup</code>	312, 329	<code>super3col</code>	293, 294, 332
<code>index</code>	7, 286, 304–306, 326	<code>super3colborder</code>	294, 332
<code>indexgroup</code>	306, 326	<code>super3colheader</code>	294, 332
<code>indexhypergroup</code>	306, 326	<code>super3colheaderborder</code>	294, 332
<code>inline</code>	321	<code>super4col</code>	295, 296, 333
<code>list</code>	7, 266–268, 321	<code>super4colborder</code>	296, 333
<code>listdotted</code>	268, 269, 322	<code>super4colheader</code>	295, 333
<code>listgroup</code>	267, 321	<code>super4colheaderborder</code>	296, 333
<code>listhypergroup</code>	267, 322	<code>superborder</code>	292, 332
<code>long</code>	269–271, 276, 280, 322, 324	<code>superheader</code>	292, 332

superheaderborder	293, 332	\glossentrydesc	264, 266–
superragged	298, 300, 330		268, 270, 271, 273, 280–282, 284, 292–
superragged3col	300, 301, 331		295, 299, 300, 302, 305–308, 311, 312, 354
superragged3colborder	301, 331	\glossentryname	264,
superragged3colheader	301, 331		266–271, 273, 280, 282, 284, 292, 293,
superragged3colheaderborder	301, 331		295, 299, 300, 302, 305–308, 310, 312, 354
superraggedborder	299, 330	\Glossentrysymbol	355
superraggedheader	299, 330	\glossentrysymbol	264,
superraggedheaderborder	300, 331		273, 284, 295, 302, 305–308, 311, 312, 355
tree	287, 306–308, 310, 326	\Gls	92, 210, 229
treegroup	287, 307, 327	\gls	91, 169, 200, 210, 229
treehypergroup	307, 327	\gls@Alphpage	175, 177
treenoname	288, 305, 308, 309, 327	\gls@alphpage	175, 177
treenonamegroup	309, 328	\gls@arabicpage	175, 177
treenonamehypergroup	309, 328	\gls@assign@desc	76, 81
glossary-hypernav package	154	\gls@assign@descplural	
glossary-list package	7, 9, 265		230, 239, 241–244, 365, 368, 369
glossary-long package	8, 269, 283, 291	\gls@assign@field	
glossary-longragged package	280		67, 71, 72, 76, 78, 80–82, 254, 255
glossary-mcols package	285	\gls@assign@firstpl	230,
glossary-super package ...	9, 269, 291, 298, 302		232–234, 236, 237, 239, 241–244, 365–369
glossary-superragged package	298	\gls@assign@plural	
glossary-tree package	9, 304		230, 232, 234, 236–239, 241–244, 365–369
\glossaryentry	178, 179, 316	\gls@assign@symbolplural	230,
glossaryentry (counter)	10, 199, 200		232–234, 236, 237, 241–244, 366, 367, 369
\glossaryentryfield		\gls@checkisacronymlist	106
	201, 321–328, 330–333, 355	\gls@checkseeallowed	62, 67, 167, 169
\glossaryentrynumbers		\gls@checkseeallowed@preambleonly ..	67
	8, 158, 183, 184, 193, 197, 198, 318	\gls@codepage	48, 166, 185
\glossaryheader	158, 191, 264, 266–	\gls@defdocnewglossaryentry	68, 88
	274, 276, 277, 280–286, 288–290, 292,	\gls@defglossaryentry	67, 68, 77
	293, 295, 298, 300, 302, 305–310, 312, 318	\gls@disablepagerefexpansion ..	175, 177
\glossarymark	38	\gls@do@addxdyattribute	43
\glossaryname	13, 33	\gls@doclearpage	40
\glossarypostamble	158, 191, 318	\gls@dostubst	110
\glossarypreamble	157, 191, 318	\gls@dotocitle	183, 195
\glossarysection	157, 191, 318	\gls@end@sanitizesort	19
glossarysubentry (counter)	10, 199, 200	\gls@endcheck	66, 67
\glossarysubentryfield		\gls@glossary	174, 178, 179
	203, 321–328, 330–333, 355	\gls@gobbleopt	58
\glossarytitle ..	157, 182, 183, 191, 195, 318	\gls@grplabel	261
\glossarytocitle	7, 13, 14, 27,	\gls@hypergroup prerun	262
	28, 31, 33, 38, 157, 183, 191, 195, 196, 318	\gls@ifnotmeasuring	86
\glossentry	83, 184, 193, 203,	\gls@inlinepostchild	263–265, 321
	264, 266–271, 273, 280, 282, 284, 292,	\gls@inlinesep	263, 264, 321
	293, 295, 299, 300, 302, 305, 307, 308, 310	\gls@inlinesubsep	263, 264, 321
\Glossentrydesc	354	\gls@islistofacronyms	15
		\gls@istfilebase	34, 166
		\gls@label	210

<code>\gls@level</code>	79, 80, 192	<code>\glscurrententrylabel</code>	181, 184
<code>\gls@noidxglossary</code>	169	<code>\glscurrentfieldvalue</code>	54, 55
<code>\gls@nosetquote</code>	77, 160, 161, 164	<code>\glscustomtext</code>	
<code>\gls@numberpage</code>	175, 177		94, 97, 98, 100, 102, 119–123, 137–144,
<code>\gls@org@glossaryentryfield</code>	184		223, 224, 231, 232, 235–238, 240, 241,
<code>\gls@org@glossarysubentryfield</code>	184		342, 345, 346, 348, 349, 351–354, 359, 360
<code>\gls@org@insert</code>	235, 238, 240	<code>\GlsDeclareNoHyperList</code>	16
<code>\gls@protected@pagefmts</code>	110, 175, 176	<code>\glsdefaulttype</code>	13,
<code>\gls@Romanpage</code>	175, 177		37, 48, 50, 56, 57, 78, 93, 103, 173, 182
<code>\gls@romanpage</code>	175, 177	<code>\glsdefmain</code>	13, 59
<code>\gls@save@numberlist</code>	8	<code>\glsdescriptionaccessdisplay</code>	
<code>\gls@suffiX</code>	36, 159, 161, 318, 320		344–346, 354, 355
<code>\gls@suffiFF</code>	36, 159, 161, 318, 320	<code>\glsdescriptionpluralaccessdisplay</code>	
<code>\gls@text</code>	102		342, 343
<code>\gls@thissty</code>	23	<code>\glsdescwidth</code>	269–272, 274,
<code>\gls@tmp</code>	173, 240		275, 277, 278, 280–285, 291–295, 297–303
<code>\gls@tmplen</code>	117, 309, 311, 312, 328, 329	<code>\glsdetoklabel</code>	
<code>\gls@tr@set@acronym@toctitle</code>	14		51–55, 63, 68, 73–77, 83, 87, 89, 90,
<code>\gls@tr@set@main@toctitle</code>	13		107, 144, 145, 152, 153, 169–171, 178,
<code>\gls@tr@set@numbers@toctitle</code>	28		184, 187, 188, 192, 193, 195–197, 199,
<code>\gls@tr@set@symbols@toctitle</code>	27		200, 202, 247–251, 310, 334, 335, 370, 371
<code>\gls@wrglossary</code>	174	<code>\glsdisplay</code>	94, 103
<code>\gls@xdystring</code>	110	<code>\glsdisplayfirst</code>	94, 103
<code>\gls@xindy@glsnumbersfalse</code>	26	<code>\glsdisplaynumberlist</code>	170
<code>\gls@xindy@glsnumberstrue</code>	25	<code>\glsdohyperlink</code>	117, 118
<code>\glsaccsupp</code>	340	<code>\glsdohypertarget</code>	117, 118
<code>\glsacronymtrue</code>	14	<code>\glsdoifexists</code>	
<code>\glsacrpluralsuffix</code>			52–54, 73–76, 86, 118–124, 136–
	31, 211, 220, 221, 225–227, 231		143, 151, 153, 170, 171, 256–260, 351–355
<code>\glsacrshortcutsfalse</code>	29	<code>\glsdoifexistsordo</code>	106, 144
<code>\glsacrshortcutstrue</code>	29	<code>\glsdoifexistsorwarn</code>	194, 201, 202
<code>\glsacspace</code>	219, 221	<code>\glsdoifnoexists</code>	67, 76
<code>\glsadd</code>	154	<code>\glsdonohyperlink</code>	108, 117, 118
<code>\glsadd options</code>		<code>\glsdosanitizesort</code>	11
counter	153	<code>\glsentryaccess</code>	340
format	153, 207	<code>\glsentrycounter</code>	206, 209
<code>\glsaddall options</code>		<code>\glsentrycounterfalse</code>	10
types	153	<code>\glsentrycounterlabel</code>	196, 200
<code>\GlsAddXdyAttribute</code>	42, 43, 314, 315	<code>\glsentrycountertrue</code>	10
<code>\GlsAddXdyCounters</code>	42, 43, 58	<code>\glsentrycurrcount</code>	89–91
<code>\glsautomakefalse</code>	26	<code>\Glsentrydesc</code>	129, 202, 354
<code>\glsautoprefix</code>	7, 195, 196	<code>\glsentrydesc</code>	
<code>\glsapscase</code>	94, 96, 98–		96, 97, 128, 129, 202, 344–346, 354
	101, 119–123, 137–143, 223, 236, 240,	<code>\glsentrydescaccess</code>	341
	342, 344, 346, 347, 349, 350, 352–354, 359	<code>\Glsentrydescplural</code>	129
<code>\glsclerpage</code>	39	<code>\glsentrydescplural</code> 94, 95, 129, 130, 342, 343	
<code>\glsclerbrace</code>	46, 158, 159, 318, 319	<code>\glsentrydescpluralaccess</code>	341
<code>\glscompositor</code>	35, 45, 161, 317, 320	<code>\Glsentryfirst</code>	92, 97, 99, 125, 345, 348
<code>\glscounter</code>	16, 29, 41, 58, 81, 108, 314		

<code>\glsentryfirst</code>	<code>\glsentryshortpl</code>
.... 91, 96, 97, 99, 125, 126, 344, 345, 348 100, 102, 138, 140, 151, 218,
<code>\glsentryfirstaccess</code>	219, 224–226, 245, 349, 351, 356, 360–363
341	<code>\glsentryshortpluralaccess</code>
<code>\Glsentryfirstplural</code> 93, 95, 98, 127, 343, 347	341
<code>\glsentryfirstplural</code>	<code>\Glsentrysymbol</code>
.... 92, 94, 95, 98, 127, 342, 343, 346, 347	130, 202, 355
<code>\glsentryfirstpluralaccess</code>	<code>\glsentrysymbol</code>
341	96, 97,
<code>\glsentryfmt</code>	130, 131, 202, 232, 236, 240, 344–346, 355
58, 59	<code>\glsentrysymbolaccess</code>
<code>\Glsentryfull</code>	341
216, 224, 226, 361, 363	<code>\Glsentrysymbolplural</code>
<code>\glsentryfull</code>	131
216, 224, 226, 360, 363	<code>\glsentrysymbolplural</code>
<code>\Glsentryfullpl</code> 216, 225, 226, 361, 363 94, 95, 131, 232, 235, 240, 342, 343
<code>\glsentryfullpl</code> 216, 225, 226, 361, 363	<code>\glsentrysymbolpluralaccess</code>
<code>\glsentryitem</code> ... 196, 264, 266–271, 273,	341
280, 282, 284, 292, 293, 295, 299, 300,	<code>\Glsentrytext</code>
302, 305, 307, 308, 310, 321–328, 330–333	96, 99, 125, 344, 348
<code>\Glsentrylong</code>	<code>\glsentrytext</code>
92, 141, 145,	96,
151, 218, 219, 224, 351, 353, 356, 359–361	97, 99, 124, 125, 152, 181, 344, 345, 347, 348
<code>\glsentrylong</code> 91, 102, 140, 142, 145,	<code>\glsentrytextaccess</code>
151, 218, 219, 221–228, 238, 351, 353–364	340
<code>\glsentrylongaccess</code>	<code>\glsentrytype</code>
341	78
<code>\Glsentrylongpl</code>	<code>\Glsentryuseri</code>
93, 143,	132
151, 218, 219, 223–225, 351, 356, 359–361	<code>\glsentryuseri</code>
<code>\glsentrylongpl</code>	132
..... 92, 102, 142, 144, 151, 218–220,	<code>\Glsentryuserii</code>
223–226, 238, 245, 351, 356, 357, 359–363	133
<code>\glsentrylongpluralaccess</code>	<code>\glsentryuseriii</code>
341	134
<code>\Glsentryname</code>	<code>\glsentryuseriii</code>
128, 202, 354	133, 134
<code>\glsentryname</code>	<code>\Glsentryuseriv</code>
128, 309, 354	134
<code>\glsentrynumberlist</code>	<code>\glsentryuseriv</code>
151, 170	134, 135
<code>\Glsentryplural</code>	<code>\Glsentryuserv</code>
95, 98, 126, 343, 347	135
<code>\glsentryplural</code>	<code>\glsentryuserv</code>
..... 94, 95, 98, 126, 342, 343, 346, 347	135
<code>\glsentrypluralaccess</code>	<code>\Glsentryuservi</code>
340	136
<code>\Glsentryprefix</code>	<code>\glsentryuservi</code>
258	136
<code>\glsentryprefix</code>	<code>\glsfieldfetch</code>
256, 259	148
<code>\Glsentryprefixfirst</code>	<code>\glsfirstaccessdisplay</code> 344, 345, 348
258	<code>\glsfirstpluralaccessdisplay</code>
<code>\glsentryprefixfirst</code> 342, 343, 346, 347
257, 259	<code>\glsfirstpluralacessdisplay</code>
<code>\Glsentryprefixfirstplural</code>	347
259	<code>\glsgenacfmt</code> 218, 219, 225, 355, 356, 361
<code>\glsentryprefixfirstplural</code> 257, 260	<code>\glsgenentryfmt</code>
<code>\Glsentryprefixplural</code> 218, 219, 224, 225, 229, 231, 234,
258	235, 238, 240, 243, 245, 355, 356, 360, 361
<code>\glsentryprefixplural</code>	<code>\glsgetgrouptitle</code>
257, 260	263, 267, 268, 286–291, 306–309, 312, 313
<code>\glsentryprevcount</code>	<code>\gls glossarymark</code>
89, 90	38
<code>\Glsentryshort</code>	<code>\gls groupheading</code> 159, 193, 264, 266–268,
100, 137,	270, 271, 273, 280, 282, 284, 286–293,
145, 219, 225, 226, 350–352, 356, 362, 363	295, 298, 300, 302, 305–310, 312, 313, 319
<code>\glsentryshort</code> 100–102, 137, 138, 145, 151,	<code>\gls groupskip</code>
216, 218–228, 349–352, 355–358, 360–364 158, 159, 193, 265, 266, 270, 271,
<code>\glsentryshortaccess</code>	273, 276, 277, 281, 282, 284, 292, 294,
341	295, 299, 301, 302, 306, 307, 309, 312, 318
<code>\Glsentryshortpl</code>	<code>\glshyperfirstfalse</code>
..... 100, 139, 220, 226, 349, 356, 362, 363	225, 361

<code>\glshyperfirsttrue</code>	24	<code>\glslongtok</code> 215, 216, 218, 219, 224, 225, 230– 239, 241–246, 355, 356, 360, 361, 364–369
<code>\glshyperlink</code>	181	<code>\glsLTpenaltycheck</code>	279
<code>\glshypernavsep</code>	263	<code>\glsmcols</code>	286–290
<code>\glshypernumber</code>	36, 209, 210	<code>\glsnameaccessdisplay</code>	354, 355
<code>\glsifhyperon</code>	105	<code>\glsnamefont</code>	201–203, 334, 335, 354
<code>\glsIfListOfAcronyms</code>	15, 16	<code>\glsnavhyperlink</code>	263
<code>\glsifplural</code>	94, 98, 100, 101, 119–123, 137–143, 223, 232, 235, 238, 240, 342, 346, 349, 350, 352–354, 359	<code>\glsnavhyperlinkname</code>	261
<code>\glsifusetranslator</code>	22, 23, 32, 33, 372	<code>\glsnavhypertarget</code> 267, 268, 286–291, 306, 308, 309, 313
<code>\glsindexonlyfirstfalse</code>	24	<code>\glsnavigation</code> 267, 268, 286–290, 306, 307, 309, 312
<code>\glsinlinedescformat</code>	264, 321	<code>\glsnextpages</code>	8, 62, 183
<code>\glsinlinedopostchild</code>	264, 321	<code>\glsnogroupskipfalse</code>	9
<code>\glsinlineemptydescformat</code>	264, 321	<code>\glsnoidxdisplayloc</code>	171, 172
<code>\glsinlinenameformat</code>	264, 321	<code>\glsnoidxdisplaylocclsthandler</code>	171
<code>\glsinlineparentchildseparator</code>	264, 321	<code>\glsnoidxloclist</code>	170, 193
<code>\glsinlinepostchild</code>	264, 321	<code>\glsnoidxloclisthandler</code>	193
<code>\glsinlineseparator</code>	264, 321	<code>\glsnoidxnumberlistloophandler</code>	171
<code>\glsinlinesubdescformat</code>	264, 321	<code>\glsnoidxstripaccents</code>	19
<code>\glsinlinesubnameformat</code>	264, 321	<code>\glsnomakeindexwarning</code>	161
<code>\glsinlinesubseparator</code>	264, 321	<code>\glsnonextpages</code>	62, 183
<code>\glsinsert</code>	94– 101, 119–123, 137–143, 223, 224, 232, 235, 236, 238, 240, 241, 342–354, 359, 360	<code>\glsnopostdotfalse</code>	9
<code>\glskeylisttok</code> 215, 216, 230–239, 241–244, 246, 364–369	<code>\glsnoxindywarning</code> ..	35, 42–44, 46–48, 155
<code>\glslabel</code>	77, 94–101, 106–108, 137– 143, 218, 219, 223–225, 231, 232, 235, 236, 238, 240, 342–351, 355, 356, 359–361	<code>\glsnumberformat</code>	152
<code>\glslabeltok</code>	215, 230–239, 241–246, 365–368	<code>\glsnumberlistloop</code>	171
<code>\glslink</code>	216, 224, 226, 231, 360, 362	<code>\glsnumbersgroupname</code>	28, 34, 205
<code>\glslink options</code>		<code>\glsnumlistlastsep</code>	152, 171
counter	104, 118, 253	<code>\glsnumlistparser</code>	152, 168
format	104, 118, 207	<code>\glsnumlistsep</code>	152, 170
hyper	104, 106, 107, 118	<code>\glsopenbrace</code>	46, 158, 159, 318, 319
local	104	<code>\glsorder</code>	25, 165–167, 189
<code>\glslinkcheckfirsthyperhook</code>	106	<code>\glsorg@endtheglossary</code>	5
<code>\glslinkpostsetkeys</code>	107	<code>\glsorg@PrintChanges</code>	5
<code>\glslinkvar</code>	105	<code>\glsorg@theglossary</code>	5
<code>\glslistdottedwidth</code>	268, 322	<code>\glspagelistwidth</code>	271, 272, 274, 275, 277, 278, 282–285, 293–295, 297, 298, 300–303
<code>\glslistgroupheaderfmt</code>	267, 268	<code>\glspatchLToutput</code>	275–278
<code>\glslistnavigationitem</code>	267, 268	<code>\glspenaltygroupskip</code>	276, 277
<code>\glslocalreset</code>	87	<code>\glspercentchar</code>	68, 69, 158, 160
<code>\glslocalunset</code>	88, 119–124	<code>\Glspl</code>	93, 229
<code>\glslongaccessdisplay</code>	351, 353–365	<code>\Glspl</code>	92, 229
<code>\glslongkey</code>	369	<code>\glspluralaccessdisplay</code>	342, 343, 346, 347
<code>\glslongpluralaccessdisplay</code> 351, 356, 357, 359–363, 365	<code>\glspluralsuffix</code> 31, 80, 81, 218–220, 356, 357, 361–363
<code>\glslongpluralkey</code>	369	<code>\glspostdescription</code> 34, 265–267, 270, 280, 281, 292, 299, 305–308, 311, 312, 321–324, 326–330, 332

<code>\glspostinline</code>	263	<code>\glstextup</code>	31, 363
<code>\glspostlinkhook</code>		<code>\glstildechar</code>	42, 158, 159
.....	106, 119–124, 137–144, 352–354	<code>\glstranslatefalse</code>	22, 23
<code>\glsprestandardsort</code>	11	<code>\glstranslatetrue</code>	23
<code>\glsreset</code>	87	<code>\glstreechildpredesc</code>	306, 307
<code>\glsresetentrycounter</code>	196, 199	<code>\glstreegroupheaderfmt</code>	
<code>\glsresetentrylist</code>	158, 191, 198, 318	286–291, 306–309, 312, 313
<code>\glsresetsubentrycounter</code>		<code>\glstreeindent</code> ..	307, 308, 310–312, 327–329
.....	196, 197, 200, 264, 321	<code>\glstreeitem</code>	286, 287, 304, 305
<code>\glssanitizesortfalse</code>	21	<code>\glstreenamebox</code>	310, 312
<code>\glssanitizesorttrue</code>	21	<code>\glstreenamefmt</code>	304–312
<code>\glssavenumberlistfalse</code>	8	<code>\glstreenavigationfmt</code>	
<code>\glssavewritesfalse</code>	27	286–290, 306, 307, 309, 312
<code>\glseeeformat</code>	157, 169, 171, 318	<code>\glstreepredesc</code>	305, 307, 308
<code>\glseeitem</code>	180	<code>\glstreesubitem</code>	286, 305
<code>\glseeitemformat</code>	181	<code>\glstreesubsubitem</code>	286, 305
<code>\glseeelastsep</code>	181	<code>\glstype</code> ..	106, 107, 119–123, 137–144, 352–354
<code>\glseeelist</code>	180	<code>\glsucmarkfalse</code>	10
<code>\glseeesep</code>	181	<code>\glsucmarktrue</code>	10
<code>\glsetexpandfield</code>	18, 20–22	<code>\glunset</code>	85, 88–90, 119–124
<code>\glsetnoexpandfield</code>	18, 20, 21	<code>\glsupacrpluralsuffix</code>	
<code>\GlsSetQuote</code>	77, 160	220, 227, 233, 237, 239, 242
<code>\glsettoctitle</code>	33, 183	<code>\GlsUseAcrEntryDisplayStyle</code>	217,
<code>\glsshortaccessdisplay</code>	220–223, 225–227, 357, 358, 361, 363, 364
.....	349–352, 355–358, 360–365	<code>\GlsUseAcrStyleDefs</code>	217,
<code>\glsshortkey</code>	369	220–223, 225–228, 357, 358, 361, 363, 364
<code>\glsshortpluralaccessdisplay</code>		<code>\glswrallowprimitivemodstrue</code>	176
.....	349, 351, 356, 360–363, 365	<code>\glswrite</code> ...	155–161, 167, 172, 173, 316–320
<code>\glsshortpluralkey</code>	369	<code>\glswritedefhook</code>	69
<code>\glsshorttok</code>		<code>\glswriteentry</code>	175
.....	215, 216, 230–239, 241–246, 365–369	<code>\glswritefiles</code>	27, 173
<code>\glssortnumberfmt</code>	12, 13	<code>\glsxindyfalse</code>	25
<code>\glsspace</code>	212	<code>\glsxindytrue</code>	26
<code>\glsstepentry</code>	196, 200		
<code>\glsstepsubentry</code>	197, 200		
<code>\glssubentrycounterfalse</code>	10		
<code>\glssubentrycounterlabel</code>	197, 200		
<code>\glssubentryitem</code>			
.....	197, 264, 266–268, 270, 271, 273,		
	281, 282, 284, 292, 293, 295, 299, 300,		
	302, 305, 307, 308, 311, 321–328, 330–333		
<code>\glssymbolaccessdisplay</code>	344–346, 355		
<code>\glssymbolpluralaccessdisplay</code> ..	342, 343		
<code>\glssymbolsgroupname</code>	27, 34, 205		
<code>\glstarget</code>	203, 204,		
	265–271, 273, 280–282, 284, 292–295,		
	299, 300, 302, 305–308, 310, 312, 321–333		
<code>\glstextaccessdisplay</code> ..	344, 345, 347, 348		
<code>\glstextformat</code>	106, 108		

H

<code>\H</code>	20
<code>\hangindent</code>	290,
.....	291, 304, 307, 308, 310, 312, 313, 326–329
<code>\hbox</code>	85, 268, 322
<code>\hfill</code>	268, 322
<code>\hline</code> ...	270–272, 274, 281–283, 285, 292–303
<code>\hsize</code>	269, 280, 291, 298
<code>\hspace</code>	304
<code>\hss</code>	268, 322
<code>\ht</code>	279
<code>\hyperdef</code>	30
<code>\hyperlink</code>	104, 117, 209
<code>hyperref</code> package	178, 181, 208, 253
<code>\hypertarget</code>	117

I

<code>\IeC</code>	19	<code>\ifglshaschildren</code>	264, 321
<code>\if</code>	110, 178, 315	<code>\ifglshasdesc</code>	264
<code>\if@endfor</code>	262	<code>\ifglshaslong</code>	91–93, 145, 218, 219, 223, 225, 238, 355, 356, 359, 361
<code>\if@gl@debug</code>	5, 17, 174	<code>\ifglshasparent</code>	187, 192, 309
<code>\if@gl@docloaded</code>	4, 13, 174	<code>\ifglshasprefix</code>	258
<code>\if@gl@isacronymlist</code>	106	<code>\ifglshasprefixfirst</code>	258
<code>\if@openright</code>	39	<code>\ifglshasprefixfirstplural</code>	258
<code>\ifbool</code>	14, 24, 27, 51, 94–96	<code>\ifglshasprefixplural</code>	258
<code>\ifboolexpr</code>	32, 56, 205	<code>\ifglshassymbol</code>	231, 235, 240, 305–308, 311, 312
<code>\ifcase</code>	6, 7, 23, 62, 195, 305, 326	<code>\ifglshyperfirst</code>	106
<code>\ifcsdef</code>	22, 33, 39, 65, 66, 72–76, 103, 174, 186, 189–191, 206, 217	<code>\ifglindexonlyfirst</code>	175
<code>\ifcseempty</code>	53, 256	<code>\ifgl@nogroupskip</code>	266, 270, 271, 273, 276, 277, 281, 282, 284, 292, 294, 295, 299, 301, 302, 306, 307, 309, 312
<code>\ifcsequal</code>	53	<code>\ifgl@nonumberlist</code>	197
<code>\ifcsstrequal</code>	76	<code>\ifgl@nopostdot</code>	9
<code>\ifcsstring</code>	75	<code>\ifgl@numberline</code>	40
<code>\ifcsundef</code>	6, 26, 29, 30, 32, 36–39, 50, 51, 58, 59, 61, 78, 81, 89, 104, 108, 109, 117, 165, 173, 181, 184, 187, 194, 195, 201, 205–208, 210, 217, 262, 320	<code>\ifgl@sanitizesort</code>	18, 21
<code>\ifdef</code>	54, 55, 63, 68, 85, 104, 144, 148, 170, 171, 211, 304	<code>\ifgl@savenumberlist</code>	65, 168, 181
<code>\ifdefempty</code>	16, 39, 50, 53–55, 59, 94, 98, 100, 168, 191–193, 215, 217, 223, 231, 235, 237, 240, 342, 346, 349, 359	<code>\ifgl@savewrites</code>	27, 165, 174
<code>\ifdefequal</code>	52–55, 65–67, 70, 79, 83, 193	<code>\ifgl@subentrycounter</code>	197, 199, 200
<code>\ifdefstrequal</code>	76	<code>\ifglstoc</code>	40
<code>\ifdefstring</code>	32, 56, 165–167, 188–190, 192–194	<code>\ifglstranslate</code>	32
<code>\ifdefvoid</code>	19, 82, 192, 193	<code>\ifgl@sucmark</code>	38
<code>\ifdim</code>	219, 279, 309	<code>\ifgl@sused</code>	91, 94–100, 106, 154, 175, 231, 235, 238, 240, 256–260, 342–349
<code>\iffalse</code>	82, 87	<code>\ifgl@swrallowprimitivemods</code>	177
<code>\IfFileExists</code>	9, 22, 23, 184	<code>\ifgl@xindy</code>	34, 35, 41–44, 46–49, 58, 83, 84, 111, 155, 161, 165, 178, 179, 184, 314–316
<code>\ifglossaryexists</code>	37, 48, 52, 165, 166	<code>\ifignoredglossary</code>	79, 82, 175
<code>\ifgl@sanitize@description</code>	20	<code>\ifin@</code>	28
<code>\ifgl@sanitize@name</code>	20	<code>\ifinlistcs</code>	190, 194
<code>\ifgl@sanitize@symbol</code>	20	<code>\ifKV@gl@link@hyper</code>	107, 108
<code>\ifgl@xindy@gl@numbers</code>	49	<code>\ifKV@gl@link@local</code>	119–124
<code>\ifgl@acrdescription</code>	244	<code>\ifmeasuring@</code>	85
<code>\ifgl@acrdua</code>	233, 240, 242, 244, 245	<code>\ifnum</code>	11, 89, 90, 192, 278, 279, 307, 308, 310, 311, 327, 328
<code>\ifgl@acrfootnote</code>	106, 244	<code>\ifstreempty</code>	321
<code>\ifgl@acrronym</code>	14	<code>\ifstrequal</code>	205
<code>\ifgl@acrshortcuts</code>	29, 229	<code>\ifthenelse</code>	21, 29, 39, 108, 167, 205, 244, 262
<code>\ifgl@acrsmalldcaps</code>	233, 234, 237, 239, 242	<code>\IfTrackedLanguage</code>	161
<code>\ifgl@acrsmaller</code>	233, 235, 237, 239	<code>\IfTrackedLanguageFileExists</code>	33, 372, 373
<code>\ifgl@automake</code>	26, 168	<code>\iftrue</code>	82, 87
<code>\ifgl@descsuppressed</code>	264	<code>\ifundef</code>	57, 68, 78, 155, 160, 167, 197
<code>\ifgl@sentrycounter</code>	196–200	<code>\ifvmode</code>	153
<code>\ifgl@sentryexists</code>	51, 52, 68, 77, 79	<code>\ifvoid</code>	279

<code>\ifx</code> ... 11–13, 15, 28, 30, 41, 42, 44, 46, 48, 49, 79–82, 84, 108, 109, 111–116, 145, 156, 159, 161–164, 173, 176, 177, 179, 180, 183, 196, 198, 206, 208, 209, 231, 233, 235, 237, 239, 240, 242, 244, 246, 247, 316–318, 320, 321, 326–329, 334, 340	<code>\longnewglossaryentry</code> 77
<code>\immediate</code> 68, 69, 91, 165, 173, 185	<code>longtable</code> package 269, 275, 280
<code>\in@</code> 28	<code>\LT@end@open</code> 279
<code>\index</code> 174	<code>\LT@err</code> 279
<code>\indexname</code> 28	<code>\LT@foot</code> 279
<code>\indexspace</code> . 266, 286–291, 306–309, 312, 313	<code>\LT@head</code> 279
<code>\input</code> 32, 93	<code>\LT@lastfoot</code> 279
<code>\inputencodingname</code> 26	<code>\LT@output</code> 279
<code>\InputIfFileExists</code> 68	
<code>\istfilename</code> . 34, 156, 160, 166, 167, 316, 319	
<code>\item</code> 266–269, 286, 287, 305, 306, 321, 322, 326	
J	M
<code>\jobname</code> 35, 68, 156, 160, 165, 166, 184, 316, 319	<code>\makeatletter</code> 68, 184
	<code>\makeatother</code> 68
	<code>\makebox</code> 268, 310, 312, 322, 328, 329
	<code>makeglossaries</code> .. 25, 35, 48, 57, 161, 167, 184
	<code>\makeglossaries</code> 6, 26, 30, 62, 168, 170, 172, 185
	<code>\makeglossary</code> 165, 167
	<code>makeindex</code> 374
	<code>makeindex</code> 10, 25, 26, 31, 34–36, 40, 56–58, 60, 84, 109, 112, 154, 157, 160, 161, 164, 173, 178, 204, 315, 316
	<code>delim_n</code> 36
	<code>delim_r</code> 36
	<code>page_compositor</code> 35
	<code>special characters</code> 111, 154
	<code>\makenoidxglossaries</code> 6, 62, 168, 172
	<code>\MakeTextUppercase</code> 4
	<code>\MakeUppercase</code> 343, 345, 352, 354
	<code>\markboth</code> 38
	<code>\mbox</code> 153, 267, 290, 291, 310, 322
	<code>memoir</code> class 174
	<code>\memUHead</code> 38
	<code>\MessageBreak</code> 33, 56, 183, 334, 372, 373
	<code>mfistuc</code> package 1
	<code>\mfistucMakeUppercase</code> 4, 38, 73, 95, 97–101, 125–136, 138, 140, 142, 144, 216, 223, 224, 226, 236, 241, 259, 260, 347–351, 359, 360, 362, 363
	<code>\midrule</code> 276, 277
	<code>\month</code> 156, 160, 316, 319
	<code>multicol</code> package 285
	N
	<code>\n</code> 160, 319
	<code>\NeedsTeXFormat</code> ... 4, 254, 314, 320, 334, 372
	<code>\new@glossaryentry</code> 67, 170
	<code>\new@ifnextchar</code> 56, 72, 73, 91, 92, 118–122, 124–143, 212–214, 256–259
	<code>\newacronym</code> 210, 215, 231, 233, 235, 237, 239, 242, 244, 246
K	
<code>\key@ifundefined</code> 70, 71	
<code>\KV@glslink@hyperfalse</code> . 104, 106, 107, 118	
<code>\KV@glslink@hypertrue</code> 104, 118	
L	
<code>\L</code> 20	
<code>\l</code> 20	
<code>\label</code> 7, 195–197, 199	
<code>\language</code> 25	
<code>\leaders</code> 268, 322	
<code>\leavevmode</code> 76, 107	
<code>\let</code> 5, 9, 11–14, 19, 20, 22, 23, 27–30, 32, 34, 43, 54–56, 65, 67, 76–82, 85, 87, 88, 90, 93, 105–108, 110, 117–124, 137–143, 145, 152, 159– 161, 164–171, 174–177, 180, 181, 183, 184, 193, 195, 198, 203, 215, 228–230, 232–244, 254, 262, 263, 279, 286, 287, 304, 305, 320, 337, 352–354, 365–369, 372	
<code>\letcs</code> 52– 55, 68, 71, 72, 75, 80, 81, 144, 145, 165, 170, 171, 186–188, 192, 193, 202, 205, 310	
<code>link text</code> 93	
<code>\listcsadd</code> 190	
<code>\listcsgadd</code> 194, 195	
<code>\listcsxadd</code> 186	
<code>\listead</code> 190	
<code>\loadglsentries</code> 93	
<code>\long</code> 76, 209	

<code>\newacronymhook</code>	215,	<code>symbol</code>	60, 61,
	231, 233, 235, 237, 239, 242, 244, 246, 364		130, 233, 234, 237, 242, 272, 295, 335–337
<code>\newacronymstyle</code>	218–223, 225–227	<code>symbolaccess</code>	339, 341
<code>\newcommand</code>	5–19, 21, 22, 24–32,	<code>symbolplural</code>	131, 336
	34–44, 46–77, 83–94, 98, 100, 102, 103,	<code>symbolpluralaccess</code>	339, 341
	105–108, 110, 111, 117–155, 161, 164–	<code>text</code>	60, 61, 118, 124, 146, 233, 237, 335
	166, 169, 172–176, 178–182, 184, 186–	<code>textaccess</code>	339, 340
	194, 197–207, 209–219, 228–252, 255–	<code>type</code>	13, 61, 93, 148
	259, 261–263, 265, 266, 278, 279, 286,	<code>user1</code>	131, 148, 337
	304, 305, 310, 320, 338–340, 355, 369–371	<code>user2</code>	132, 149
<code>\newcount</code>	12, 65	<code>user3</code>	133, 149
<code>\newcounter</code>	196–199	<code>user4</code>	134, 149
<code>\newenvironment</code>	201	<code>user5</code>	135, 149
<code>\newglossary</code>	13, 14, 27, 28, 58, 167	<code>user6</code>	135, 150, 337
<code>\newglossaryentry</code>	28, 64, 67, 88, 215, 230,	<code>\newglossarystyle</code>	
	232, 234, 236, 238, 241, 243, 245, 365–368	 263, 266–278, 280–303, 305–310, 312
<code>\newglossaryentry options</code>		<code>\newif</code>	4, 15, 22, 25, 176
<code>access</code>	337, 338	<code>\newlength</code>	117, 269, 280, 291, 298, 308
<code>counter</code>	61	<code>\newrobustcmd</code>	
<code>description</code> 67, 68, 72, 73, 91, 92, 106, 118–143,
. 24, 60, 64, 67, 77, 128, 146, 211, 239, 336			145–151, 153, 154, 211–214, 255–259, 309
<code>descriptionaccess</code>	339, 341	<code>\newterm</code>	28
<code>descriptionplural</code>	129, 336	<code>\newtoks</code>	111, 165, 215
<code>descriptionpluralaccess</code>	339, 341	<code>\newwrite</code>	68, 155, 160, 165, 167
<code>first</code>	61, 80, 118, 125, 147, 237, 242, 335	<code>ngerman package</code>	161
<code>firstaccess</code>	339, 341	<code>\noalign</code>	279
<code>firstplural</code>	61, 126, 147, 336	<code>\nobreak</code>	267, 279, 322
<code>firstpluralaccess</code>	339, 341	<code>\noexpand</code> .	15, 30, 41, 43, 81, 82, 103, 108–
<code>format</code>	156		110, 116, 117, 152, 162–166, 168, 176,
<code>long</code>	100, 150, 336		177, 181, 184, 185, 187, 188, 201, 203,
<code>longaccess</code>	340, 341		210, 215, 216, 230, 232, 234, 236, 238,
<code>longplural</code>	150, 336		241, 243, 245, 246, 314, 334, 335, 365–369
<code>longpluralaccess</code>	340, 341	<code>\nohyperpage</code>	208
<code>name</code>	60, 64, 67, 77, 127, 145, 181, 335	<code>\noindent</code>	203, 287–290, 307–309
<code>nonumberlist</code>	62, 63	<code>\noist</code>	319, 320
<code>parent</code>	62, 67	<code>\nopostdesc</code>	28, 34, 76, 183, 321
<code>plural</code>	61, 80, 126, 336	<code>\normalbaselineskip</code>	278, 279
<code>pluralaccess</code>	339, 340	<code>\nr</code>	6, 7, 23, 62, 195
<code>prefix</code>	254	<code>\ns@ACRfull</code>	213
<code>prefixfirst</code>	254	<code>\ns@Acrfull</code>	212
<code>prefixfirstplural</code>	255	<code>\ns@acrfull</code>	212
<code>prefixplural</code>	255	<code>\ns@ACRfullpl</code>	214
<code>see</code>	5, 8, 62, 67, 167, 169	<code>\ns@Acrfullpl</code>	214
<code>short</code>	100, 150, 336	<code>\ns@acrfullpl</code>	213
<code>shortaccess</code>	339, 341	<code>\ns@ACRlong</code>	141
<code>shortplural</code>	150, 336	<code>\ns@Acrlong</code>	140, 141
<code>shortpluralaccess</code>	339, 341	<code>\ns@acrlong</code>	140
<code>sort</code>	60, 148, 204	<code>\ns@ACRlongpl</code>	143
		<code>\ns@Acrlongpl</code>	142, 143

`\ns@acrlongpl` 142
`\ns@ACRshort` 137, 138
`\ns@Acrshort` 137
`\ns@acrshort` 136
`\ns@ACRshortpl` 139
`\ns@Acrshortpl` 139
`\ns@acrshortpl` 138
`\ns@newglossary` 57
`\null` 110–117, 162–164, 184
`\number` 11, 80, 90, 175, 177, 203, 335
`\numberline` 40
`\numexpr` 90

O

`\O` 20
`\o` 20
`\OE` 20
`\oe` 20
`\openout` 68, 156, 160, 165, 316, 319
`\OR` 244
`\or` 6, 7, 23, 195, 196, 305, 326
`\org@glossaryentrynumbers` 183, 198
`\org@glossarytitle` 183
`\org@glspostdescription` 34
`\org@ifKV@glslink@hyper` 107, 108
`\orgAlph` 177
`\orgalph` 177
`\orgarabic` 177
`\orgnumber` 177
`\orgRoman` 177
`\orgromannumeral` 177
`\orgthe` 177
`\outputpenalty` 278, 279

P

`\p@` 266, 285, 304
`\p@gl@hyp@opt` 105
package options:
 `acronym` 13, 14, 30, 182, 211
 `true` 14
 `counter` 16
 `description` 237
 `dua` 235, 237
 `entrycounter` 196, 198
 `true` 10
 `footnote` 119–123, 233, 235, 237, 239
 `hyperfirst`
 `false` 119–123
 `indexonlyfirst` 381
 `makeindex` 158, 253

`nogroupskip` 270, 271, 273, 276, 277,
 281, 282, 284, 292, 294, 295, 299, 301, 302
`nolist` 246
`nolong` 246, 269
`nomain` 13
`nonumberlist` 8
`nosuper` 246
`notree` 247
`nowarn` 5
`numberline` 6
`sanitize` 20, 60, 145, 146
`sanitizesort` 17
`savewrites` 27, 378
 `false` 165
 `true` 166, 173
`section` 6, 38
`sort`
 `def` 10, 11
 `standard` 10
 `use` 10, 11
`style` 7, 246, 247
`subentrycounter` 196, 199
`toc` 6
 `true` 6
`translate` 23
 `false` 22
`translator` 22
`xindy` 25, 26, 158, 253
`\PackageError` 5,
 6, 26, 30, 42, 48, 51, 52, 56, 62, 64, 65,
 71–76, 78–80, 88, 104, 144, 164, 165,
 167, 170, 172, 189–191, 195, 197, 206,
 207, 217, 233–235, 240, 242, 243, 320, 342
`\PackageInfo` 5, 165, 174
`\PackageWarning` 5, 17
`\PackageWarningNoLine` .. 5, 17, 33, 372, 373
`\pagegoal` 279
`\pagelistname` 33, 272–
 274, 276, 277, 283–285, 294–297, 301–303
`\par` 34, 203, 204, 266, 267, 285,
 287–291, 304, 305, 307–313, 322, 327–329
`\parindent` 286–291, 305–308, 310–313, 327–329
`\parskip` 286–289, 305, 306, 308
`\PassOptionsToPackage` 254, 334
`\penalty` 279
`\phantomsection` 39
`polyglossia package` 22, 32
`\printglossaries` 168
`\printglossary` .. 14, 17, 27, 28, 168, 182, 197

<code>\printglossary options</code>	290, 291, 304, 305, 307–313, 315, 318–320, 326–329, 337, 352, 353, 365–367, 369
<code>entrycounter</code>	196
<code>nogroupskip</code>	196
<code>nonumberlist</code>	197
<code>nopostdot</code>	196
<code>numberedsection</code>	195
<code>style</code>	195
<code>subentrycounter</code>	196
<code>title</code>	195
<code>toctitle</code>	195
<code>type</code>	13, 181, 195
<code>\printindex</code>	28
<code>\printnoidxglossaries</code>	169
<code>\printnoidxglossary</code>	169, 172, 182, 189, 190, 197
<code>\printnoidxglossary options</code>	
<code>sort</code>	197
<code>\printnumbers</code>	28
<code>\printsymbols</code>	27
<code>\ProcessOptions</code>	254, 334
<code>\ProcessOptionsX</code>	29
<code>\protect</code>	40, 102, 218–220, 225, 226, 232, 236, 238, 351, 355, 356, 361, 362
<code>\protected@edef</code>	7, 43, 44, 47, 49, 79, 82, 84, 94–96, 102, 109, 152, 174, 177, 196, 201, 203, 206, 215, 240, 245, 255, 261, 315, 316, 334, 335, 340
<code>\protected@write</code>	56, 57, 156, 158, 167, 169, 172, 174, 181, 261, 316
<code>\protected@xdef</code>	11–13, 15, 19, 66, 84, 85, 177, 338
<code>\providecommand</code>	14, 30, 31, 38, 56, 91, 118, 158, 167, 169, 185, 201, 203, 266, 285, 304
<code>\ProvidesFile</code>	32
<code>\ProvidesPackage</code>	4, 254, 261, 263, 265, 269, 275, 280, 285, 291, 298, 304, 314, 320, 334, 372
R	
<code>\r</code>	19
<code>\raggedright</code>	278, 280–285, 298–303
<code>\raisebox</code>	117
<code>\ref</code>	200
<code>\refstepcounter</code>	196, 197, 199
<code>\relax</code>	7, 9, 12–14, 22, 23, 29, 43, 56, 61, 62, 66, 79, 82, 85, 89, 90, 105, 106, 108, 110–116, 145, 158–164, 167, 169–171, 175, 177, 178, 180, 183, 184, 192, 195, 205, 206, 247, 262, 266, 278, 285, 290, 291, 304, 305, 307–313, 315, 318–320, 326–329, 337, 352, 353, 365–367, 369
<code>\renewacronymstyle</code>	355–359, 361, 363, 364
<code>\renewcommand</code>	4–10, 13, 14, 16, 17, 21, 23–27, 29, 32–36, 48, 59, 63, 76, 88–90, 152, 153, 155, 161, 162, 167–171, 174, 184, 185, 195–197, 215, 216, 218–228, 231, 233, 235, 237, 239, 242, 244, 246, 264–274, 276, 277, 279–295, 298–302, 305–315, 320–328, 330–333, 337, 342, 346, 349, 351, 354–358, 360–368
<code>\renewenvironment</code> ..	201, 263, 266, 269–275, 277, 278, 280–303, 305, 306, 308, 310
<code>\RequireGlossariesLang</code>	33, 372, 373
<code>\RequirePackage</code>	4, 8, 9, 22, 23, 29, 32, 246, 253, 254, 269, 275, 280, 285, 291, 298, 335, 372
<code>\restorecounters@</code>	109
<code>\romannumeral</code>	176, 177, 310, 311, 328
S	
<code>\s@gl@s@hyp@opt</code>	105
<code>\s@newglossary</code>	57
<code>\savecounters@</code>	108
<code>\seenname</code>	180
<code>\SetAcronymStyle</code>	24, 25
<code>\setbool</code>	21
<code>\setbox</code>	279
<code>\setcounter</code>	196, 197, 199
<code>\SetCustomDisplayStyle</code>	246
<code>\SetDefaultAcronymDisplayStyle</code>	230, 231
<code>\SetDefaultAcronymStyle</code>	244
<code>\SetDescriptionAcronymDisplayStyle</code>	237
<code>\SetDescriptionAcronymStyle</code>	244
<code>\SetDescriptionDUAAcronymDisplayStyle</code>	235
<code>\SetDescriptionDUAAcronymStyle</code>	244
<code>\SetDescriptionFootnoteAcronymDisplayStyle</code>	233
<code>\SetDescriptionFootnoteAcronymStyle</code>	244
<code>\SetDUADisplayStyle</code>	244
<code>\SetDUASStyle</code>	245
<code>\setentrycounter</code>	42, 158, 194, 314
<code>\SetFootnoteAcronymDisplayStyle</code> ...	239
<code>\SetFootnoteAcronymStyle</code>	244
<code>\SetGenericNewAcronym</code>	217
<code>\setglossarystyle</code>	183, 206, 247, 267–278, 281, 283–290, 292–294, 296, 297, 299–301, 303, 306, 307, 309, 312

<code>\setglossentrycompatibility</code> ...	195, 206	<code>textcase</code> package	4
<code>\setkeys</code> ..	22, 26, 29, 38, 78, 107, 153, 183, 215, 231, 233, 235, 237, 239, 242, 244, 246	<code>\textit</code>	210
<code>\setlength</code>	269, 280, 286–289, 291, 298, 305, 306, 308, 311, 312, 328, 329	<code>\textmd</code>	209
<code>\SetSmallAcronymDisplayStyle</code>	242	<code>\textrm</code>	209
<code>\SetSmallAcronymStyle</code>	245	<code>\textsc</code> ..	210, 220, 226, 233, 237, 239, 242, 363
<code>\settoheight</code>	117	<code>\textsf</code>	209
<code>\settowidth</code>	219, 309–311, 328	<code>\textsl</code>	210
<code>\sfcode</code>	9	<code>\textsmaller</code>	220, 221, 227, 233, 237, 239, 242, 363
<code>\show</code>	247–252, 370, 371	<code>\texttt</code>	209
<code>\SmallNewAcronymDef</code>	242	<code>\textulc</code>	211
<code>\space</code> ..	6, 26, 30, 42, 46, 49, 62, 64, 88, 89, 91–93, 102, 103, 105, 151, 156–160, 164, 166–170, 172, 181, 183, 185, 196, 197, 200, 206, 212, 218–228, 236, 240, 263, 265–267, 270, 280, 281, 292, 299, 304– 308, 310–312, 314, 315, 317–319, 321– 324, 326–330, 332, 351, 355–358, 360–365	<code>\textup</code>	210, 211
<code>\spacefactor</code>	9	<code>\th</code>	20
<code>\SS</code>	20	<code>\the</code>	30, 33, 42, 47, 49, 57, 111–116, 156, 160, 162–164, 173, 174, 177, 181, 192, 201, 203, 209, 215, 216, 218, 219, 224, 225, 230, 232, 234, 236, 238, 241, 243, 245, 246, 314, 316, 319, 335, 355, 356, 360, 361, 364–369
<code>\ss</code>	20	<code>\the@numberlist</code>	152
<code>\string</code>	6, 17, 26, 30, 40–42, 44–47, 49, 56, 57, 62, 64, 68, 69, 72, 73, 83, 84, 88, 89, 91–93, 103, 105, 109, 110, 112–114, 116, 151, 154–161, 163–165, 167–170, 172, 178, 179, 183, 185, 189, 190, 197, 203, 206, 261, 314–320	<code>\theglossary</code>	5
<code>\strut</code>	204, 266–268, 270, 271, 273, 281, 282, 284, 292, 294, 295, 299, 300, 302, 308, 321–325, 327, 330–333	<code>\theglossaryentry</code>	196, 198, 200
<code>\subglossentry</code> ..	83, 184, 192, 203, 264, 266– 268, 270, 271, 273, 281, 282, 284, 292, 293, 295, 299, 300, 302, 305, 307, 308, 311	<code>\theglossarysubentry</code>	197, 199, 200
<code>\subitem</code>	286, 304, 305, 326	<code>\theglentrycounter</code> ..	108, 109, 177, 315, 316
<code>\subsubitem</code>	286, 304–306, 326	<code>\theH</code>	179
<code>supertabular</code> package	9, 246, 291, 298	<code>\theHglossaryentry</code>	196, 198
<code>\symbolname</code>	33, 273, 274, 277, 284, 285, 296, 297, 303	<code>\theHglossarysubentry</code>	197, 199
		<code>\theHglentrycounter</code>	109, 177
		<code>\thesection</code>	30
		<code>\this@dialect</code>	33, 372, 373
		<code>\toks@</code> ..	30, 32, 33, 42, 47, 49, 57, 111–116, 162–164, 181, 201, 203, 209, 314, 334, 335
		<code>\toprule</code>	276, 277
		<code>tracklang</code> package	32, 372
		<code>\trans@languages</code>	32
		<code>\translate</code>	33, 34
		<code>\translatelet</code>	13, 14, 27, 28
		<code>translator</code> package ..	13, 14, 22, 27, 28, 32, 33, 181
		<code>\TX@trial</code>	85

T

<code>\t</code>	19
<code>\tablehead</code>	291–303
<code>\tabletail</code>	291–303
<code>\tabularnewline</code> ..	270–274, 276, 277, 280– 285, 292–297, 299–303, 324, 325, 330, 331
<code>\texorpdfstring</code>	148
<code>\textbar</code>	263
<code>\textbf</code>	203, 209, 304, 326–329

U

<code>\u</code>	19
<code>\uccode</code>	192
<code>\undef</code>	63, 182
<code>\unskip</code>	76, 268, 322
<code>\unvbox</code>	279
<code>\usedictionary</code>	32
<code>\usepackage</code>	189, 190

V		\xatlevel@	108
\v	20	\xcapitalisewords	148
\val	6, 7, 23, 62, 195	\xdef	73, 79, 80, 82, 184, 262
\vbox	279	\xglaccsupp	340
\vsize	279	\xifinlistcs	186, 187, 190
\vskip	266, 278, 279, 285, 304	xindy	374
\vss	279	xindy 10, 25, 26, 34, 35, 40, 44, 46–49, 84, 115, 116, 155, 157, 173, 178, 184, 204, 252, 315	
W		\xmakefirstuc	95, 96, 102, 144, 146, 255
\warn@nomakeglossaries	167, 169, 170	\xspace	210
\warn@noprintglossary	167, 169, 184	xspace package	4, 210
\write	68, 69, 91, 156–161, 165, 166, 169, 173, 185, 316–320	Y	
\writeist	164, 165, 320	\year	156, 160, 316, 319
X		Z	
\x	209	\z@	279